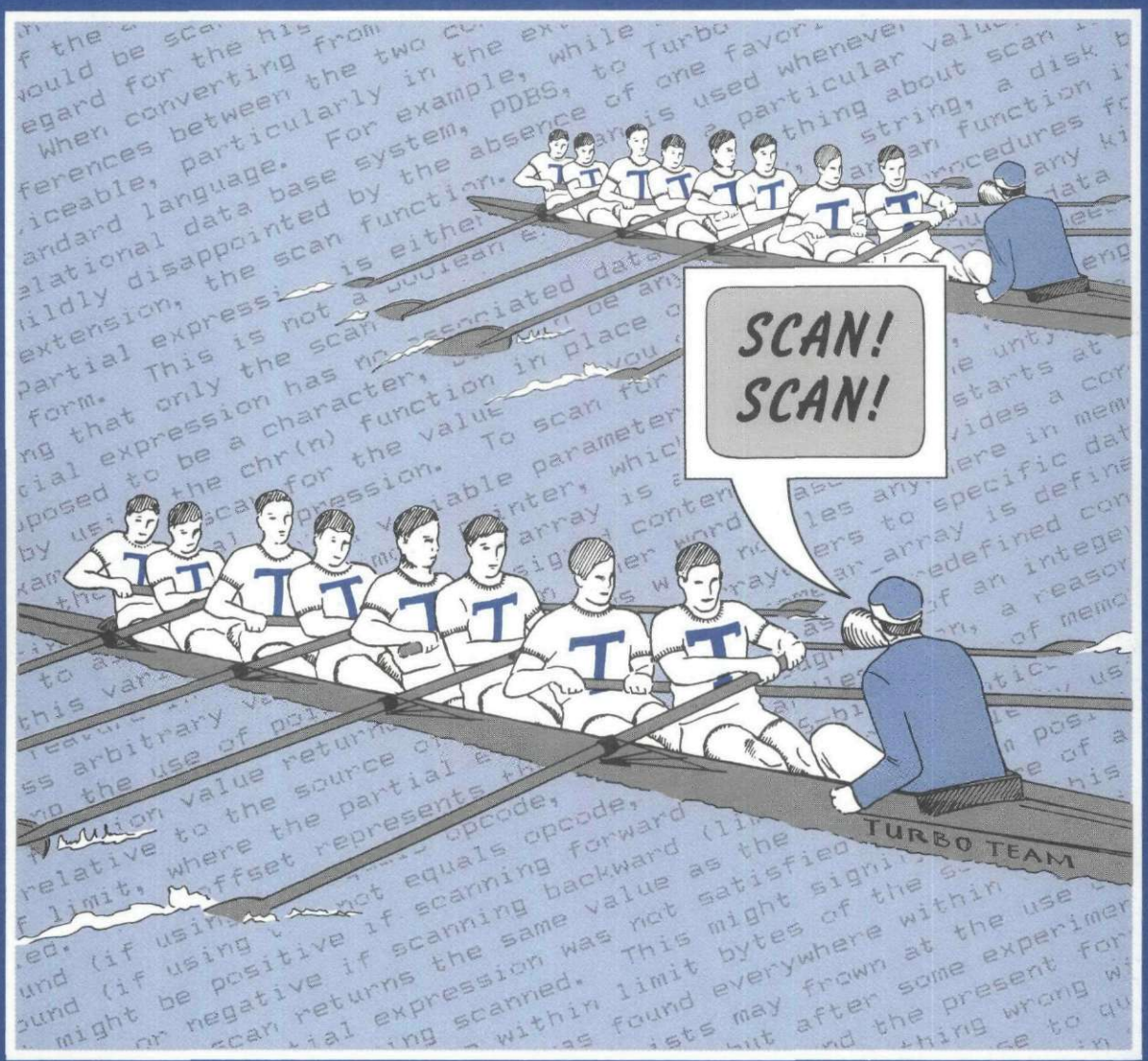
$4.00

programmer's journal

The Resource Journal for IBM PC Programmers

VOLUME 3, NUMBER 2



SCANNING IN TURBO PASCAL

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

POWER TOOLS

from *Phoenix*

You are a serious programmer. Your income depends on developing software solutions for complex problems. Fast. Error free. You need powerful tools for your compiler. That's what we sell.

PLINK ™ 86
OVERLAY LINKAGE EDITOR

- overlays for modules are automatically loaded during execution
- generates symbol tables for PFIX 86 PLUS
- accepts unlimited number of files
- capacity for over 35,000 symbol, commonblock and variable names
- change overlay structure without recompiling
- sorted memory map reports
- user defined width/height for reports
- define symbols as absolute addresses or offsets from other symbols
- over loader is user configurable

List: $395

PFIX ™ 86 PLUS
DYNAMIC SYMBOLIC DEBUGGER

- debugs PLINK 86 overlays
- view source code while debugging
- set temporary or permanent breakpoints
- advanced tracing and trapping capability
- in-line assembler for temporary patches
- preserves program screen and supports dual CRT configurations
- multiple windows to view code, data, breakpoints, register/stack contents, simultaneously
- supports 8087 numeric coprocessor

List: $395

PMATE ™
TEXT EDITOR PROCESSOR

- automatic disk buffering
- 10 auxiliary buffers that can be edited
- garbage stack
- user customization
- 120 comand macro language
- store macros permanently
- arithmetic and logical operations
- 100 numeric variables
- automatic formatting with indent and word wrap
- insert, overwrite and command modes
- horizontal/vertical scrolling

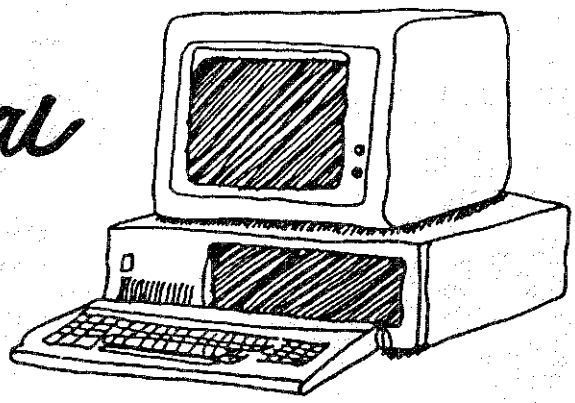
List: $225

[PC]² We Sell Power

Phoenix Computer Products Corporation
1416 Providence Hwy., Suite 220
Norwood, MA 02062
800-344-7200 (Mass.) 617-762-5030
Visa • Mastercard • American Express

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

programmer's journal



CONTENTS

HOUSEKEEPING 2

LETTERS 3

PJ FORUM: UNIX! (AGAIN) by FNE 5
Excerpts from DTACK GROUNDED The 68000/32081 Systems Journal

REDIRECTIONS: ARTIFICIAL INTELLIGENCE by Frank Greco 11
A rash of new entries in the AI marketplace

C: DOCTOR C'S TIPS — POINTERS TO FUNCTIONS by Rex Jaeschke 20
More sophisticated pointer usage

ASM: THE ASK/BATCH FILE UTILITY by Dan Rollins 22
Ask the user and get the answer via a Batch file

BASIC: INDEX FILES (Part II) by Ronald Moak 26
The second half of the saga

PASCAL: SCANNING IN TURBO PASCAL by Tom Swan 30
A routine for all seasons

BASIC: CALENDAR DATE ROUTINES IN BASIC by Bernard H. Robinson, Jr. 33
Routines that work until February of the year 4000

TOOLS: CALCULATE RECORD LAYOUTS, AUTOMATICALLY !! by J. Patrick Foley 38
Spreadsheets do all the work for your record layout shuffling

LANGUAGES: ONE PROGRAMMER'S OPINION by Bill Salkin 42
Hit the trenches and dig in for your choice

REVIEW: BLAISE'S CTOOLS by Rex Jaeschke 45
A Library for the C software developer

MISC 47

AD INDEX 47

ISSN#0747-5861

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HOUSEKEEPING

It's high time you took a break from programming. I mean it. Take yourself, your girlfriend or spouse, your kids and the dog, throw them all in the car and go to the beach. And don't play any video games or pick up a computer mag on the way there or back. Enjoy a week of relaxation and unwind from the "data acquisition" state of mind we are always in. I'm serious.

Pause for a long hard look at what you are doing. Software development and consulting are just like the polar ice cap. It may be getting bigger or smaller (who knows?), but the sheer mass is more than the whole lot of us can handle. Even Earl Scruggs, the ultimate banjo player, has his "time off" months. He'll stash his 5-string in the case and play electric bass with his sons for a while. He returns with a new outlook.

If chocolate cream pie happens to be your favorite dessert, then order only banana cream pie this week. After the break, you will at worst have a renewed appreciation for your chosen vices, and at best you will find that there are other pleasures that bring you the same enjoyment.

You remember the book *The West Ridge* by Thomas Hornbein? It chronicled the American ascent of the West Ridge of Mount Everest. The first chapter begins on the first day of their walk from Kathmandu to Base Camp. The two prominent members of the expedition, Hornbein and Unsoeld, both expressed no particular interest in climbing the mountain. Each one had decided to join the trek because the other had agreed to go. The coming months of hiking and climbing were merely a suitable diversion while they reevaluated the current direction of their lives. They liked to take a total break every five years or so and reexamine their goals. It just so happened that Everest expeditions always seemed to coincide with their personal plans.

So pick a ptomaine place on the map — Jamaica, the Canadian Rockies, Baja, or Tomaine Tommy's Asian Diner & Lakeside Motel — and go for some long walks. Be gone for a month if you can. Yeah, they will have announced five new PC clones, and Intel will have begun work on the 80586 chip, but it is really just more of the same. When was the last time you took a break? What if all of us the entire computer industry, took the month of April off? Not such a bad idea.

Now, where are my beach sandals? I think they are in the garage.

— Greg □

programmer's journal

Publisher:	Elizabeth Oakley
Editor:	Greg Estes
Managing Editor:	Louise Shimmel
Copy Editor:	Marilyn Reaves
Production Manager:	Christine Baldwin
Production Assistant:	Jennifer Mullock
Cover Design:	Marilyn Reaves
Illustrations:	Julia King
Advertising Director:	Louise Shimmel
Contributing Writers:	Ron Moak Frank Greco Tom Swan Dan Rollins Rex Jaeschke Bill Salkin FNE J. Patrick Foley Warren Hillman Bernard H. Robinson Jr.
Published by:	Avant-Garde Publishing Corp.
Business Office:	Avant-Garde Publishing Corp. 1907 Garden Ave. Eugene, OR 97403 (503) 345-3043
Editorial Office:	2765 Potter St. Eugene, OR 97405 (503) 687-0398
Subscription Inquiries:	Programmer's Journal P.O. Box 3000 Dept. EE Denville, NJ 07834 (503) 345-3043
Postmaster send address changes to:	Programmer's Journal P.O. Box 3000 Dept. EE Denville, NJ 07834

IBM and Personal Computer are trademarks of International Business Machines, Armonk, New York; dBASE II is a trademark of Ashton-Tate.
Published bi-monthly.

Copyright © 1985
by Avant-Garde
Publishing Corp.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

COMMAND.COM PATCH FOR PIPES

One sore point concerning PC-DOS Versions 2.00+ is that pipes are implemented as temporary files instead of as true connections between concurrently executing programs. The problem becomes especially severe when you're working on a floppy disk, since the disk arm moves around a great deal. One way to solve this problem is to set up a RAMDISK and use it as the default drive, to force PC-DOS to create pipe files there. The disadvantage to this is that drive specifiers must be used to access any other drive.

A quick solution to this problem exists, however. It is possible to patch COMMAND.COM to force it to use a specific drive for pipe files without regard to the current default setting. When you create a pipe file, COMMAND.COM issues the DOS service call 19 (returns default drive code in AL) and puts the pipe files on whatever drive is returned. The patch replaces the INT 21 instruction with a MOV AL,drive#. To install the patch, use a backup copy of your DOS disk and follow the procedure below:

```
A> DEBUG COMMAND.COM          /* Open the door....
- A addr                      /* See below for "addr"
ssss:addr MOV AL,drive#      /* Force to our drive
ssss:addr+2                  /* Hit return here
- W                           /* Write back to disk
- Q                           /* End Debug

PC-DOS   addr
2.00     1867
2.10     187A
3.00     2D04

drive#
A:  0
B:  1
C:  2
etc
```

Happy plumbing,

George W. Smythe
Jericho, NY

BUG IN VOLUME 2, #6, DATE ROUTINES

While showing a friend the Calendar Date Function routines from our article in Vol. 2 #6 we found a pesky little bug.

This bug shows up if a calculated date happens to be January 31. Enter 1/1/85 for the first date, no second date, and 30 for the number of days. The display for the second day should show "Thursday, January 31, 1985, is Day # 31 of 1985." What it does show is "Thursday, February 0 ..."!

The bug lurks in the subroutine to calculate the date of a given day number, and can be swatted quite easily. As published, the last figure in line 2805 is 30.6. Change it to 30.6001 so that the last term in the line now reads:

FIX(NYEAR/100)-7)))/30.6001)

Our apologies to those who might have been inconvenienced by this.

Sincerely,

Jehangir Gazdar
Roomy R. Shroff
Bombay, India

A WORD ABOUT PROGRAMMING STYLE

Programmers are for the most part a rather independent lot. Most of us have our own ideas about things and above all else know we can write a better program than anyone else, given the time (always in short supply, of course). Each of us has a "favorite" programming language and just hates to program in another, especially if someone else says we must. But, just because we are independent to a fault doesn't mean that we, as programmers, can not or will not learn from others.

When we first learn to program, (in whatever language), we all copy the style and methods of the teacher, book, article,... But as each of us progresses along the yellow brick programming road, the truly "innovative" programmer makes his/her own path. This is not surprising, given the independent nature of a programmer.

Which brings me to a point about programming style. I am the first to admit that probably no one else will like every facet of my style, although some may like parts of it. I am also certain that it doesn't matter, because my style is my own. It follows my thought patterns in a general sort of way, and that makes it easier for me to program.

My style has developed over time along with my level of expertise. Whether the development is forward or backward is not for me to say. But this I will say: I make a special effort to learn from other programmers. I read and study any code I get my hands on. I look at the style. I search for new algorithms that do things differently from ones I have developed. When I see something that is more efficient than my method or fits closely my way of programming, I rethink my method and

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MEMO: *C Programmers*

QUIT WORKING SO HARD.

These people have quit working so hard: IBM, Honeywell, Control Data, GE, Lotus, Hospitals, Universities & Government Aerospace.

THE GREENLEAF FUNCTIONS™

THE library of C FUNCTIONS that probably has just what you need . . . **TODAY!**

. . . already has what you're working to re-invent
. . . already has over 200 functions for the IBM PC, XT, AT, and compatibles
. . . already complete . . . already tested . . . on the shelf
. . . already has demo programs and source code
. . . already compatible with all popular compilers
. . . already supports all memory models, DOS 1.1, 2.0, 2.1
. . . already optimized (parts in assembler) for speed and density
. . . already in use by thousands of customers worldwide
. . . already available from stock (your dealer probably has it)
. . . It's called the **GREENLEAF FUNCTIONS.**

Sorry you didn't know this sooner? Just order a copy and then take a break — we did the hard work. Already.

THE GREENLEAF FUNCTIONS GENERAL LIBRARY: Over 200 functions in C and assembler. Strength in DOS, video, string, printer, async, and system interface. All DOS 1 and 2 functions are in assembler for speed. All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Polled mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library.**) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions.** Simply the finest C library (and the most extensive). All ready for you. From Greenleaf Software.

. . . **Specify compiler** when ordering. Add $7.00 each for UPS second-day air. MasterCard, VISA, check, or P.O.

- **Compilers:**
  - CI C86 . . . . . $349
  - Lattice $395
  - Mark Williams . . . $475
- **General Libraries $175**
  (Lattice, Microsoft, Mark Williams, CI C86)
  (DeSmet C) $150
- **Comm Library $160**

GREENLEAF SOFTWARE©

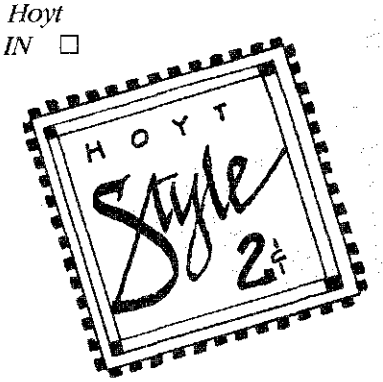
GREENLEAF SOFTWARE, INC.
2101 HICKORY DRIVE ◆ CARROLLTON, TX 75006 ◆ (214) 446-8641

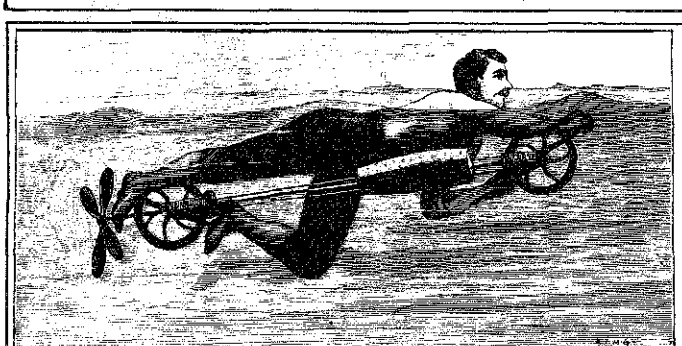
tend to adopt the "new" into my style or programming techniques. In this way I feel that I am learning from others and not just relying on my own necessarily limited experiences.

I think it is a mistake to try and force one's programming to fit someone else's idea of "good form" for the sake of style alone. A programmer's style is his/her own and should not be designed by someone else. But, if you run across a good idea developed by another, don't be afraid to admit that it's better than yours. Use it as a starting place and build on it. This is not copyright infringe-ment (you can not copyright an idea), but rather imitation. Imitation is the highest compliment, for through imitation, an idea lives on.

Don't make your style fit someone else's. Rather, let your style be one that fits the patterns of your mind and continues to develop and change as you learn. Be inventive. Take what others have done in the past and build on it. The computer field did not reach its present level because people were content to duplicate what was done previously. Let's make sure that we, as programmers, keep pushing the limits of our field to new levels.

Michael L. Hoyt
Solsberry, IN □





Software Deadline Enhancement Tools: #6
Chief Programmer's Post-Deadline
Hot Water Gear

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

UNIX! (AGAIN)

Excerpts from DTACK GROUNDED
The 68000/32081 Systems Journal

by FNE, Copyright (c) 1984 by Digital Acoustics, Inc.

Way back in early 1982, there was a furious debate about which operating system was going to win out in the 16-bit personal computing arena. Soon the UNIX adherents were shouting louder than most other folk. They started making some pronouncements which seemed to us at the time to be remarkably fatuous, so we placed a two-page writeup (pp. 2-3) in the June '82 issue of this rag. That writeup was intended to clarify a few items for our readers, and we figured that would end it.

WRONG!

It seems that the UNIX arena is public-relations-driven, which means that fantasy pretends to be reality and crass facts are ignored. A limited few UNIX gurus whipped themselves into a frenzy of absurdly overstated expectations during the summer of 1982, and the well-oiled UNIX public relations departments saw to it that those expectations got lots of ink. Dozens and dozens of minicomputer types started huddling with venture capitalists to grab a piece of the huge UNIX market which, obviously, was poised for explosive growth! These expectations climaxed in the Fall of 1982 when UNIX guru Jean Yates, frothing with excitement, jumped up on the nearest table (metaphorically speaking) and shouted to the world that 600,000 UNIX systems would be sold in 1983!

An amazing number of otherwise rational persons swallowed that prediction hook, line, and sinker. MINI-MICRO SYSTEMS magazine pointed out that Microsoft would gross $2 BILLION in 1983 in XENIX licenses. It was soon reported that there were ninety, count'em, 90, 68000-based UNIX startups in Silicon Valley alone. Far from laying the question of UNIX to rest in the June '82 issue, we felt it necessary to counterbalance in these pages the effluvium of toro poo-poo from other publications. This meant that everyone was out of step except Johnny, uh, Felgercarb. And it meant that Felgercarb took an awful lot of lak from (some) DTACK readers over his view of UNIX.

As we have pointed out before, this (personal computer) industry is highly susceptible to the 'Chicken Little' effect. The henhouse emptied out and all the chickens ran off behind Jean Yates squawking, "UNIX is coming! UNIX is coming!" Your FNE received a number of letters pointing out that we would join the rush if only we knew more about UNIX (we STILL get such letters).

It slowly became evident in 1984 that UNIX had sold somewhat fewer than 600,000 systems in 1983, and the charge of the chickens ran out of steam. But nobody, especially those opinion-leaders whose pronouncements are amplified by the UNIX public relation offices, likes to admit they goofed. Jean Yates muttered that if 600,000 had been sold — and even invented a lot of non-extant UNIX vendors to bolster that claim. Others looked for new champions — IBM or AT&T (Fortune having fallen on its derriere).

We still giggle when we remember the publisher/editor who called us right after IBM announced PC/IX and triumphantly proclaimed that IBM had lined up behind UNIX and that, contrary to what we had been telling our readers, UNIX would now inevitably emerge victorious! We replied in print by questioning whether sales of PC/IX at $900/copy would pay for the paper clips in IBM's Seattle office. And then there was System V and then there was AT&T's entry into the hardware arena and... Different UNIX gurus desperately grasped onto one or the other of these announcements (UNIX is VERY big on announcements) as evidence of UNIX's inevitable triumph.

A funny thing became evident: the UNIX marketplace WAS expanding — but at about the same rate as the rest of the small computer marketplace. If but a tenth of the nonsense pumped out by the collective UNIX public relations departments were true, UNIX would today be the only operating system used on any computer within the ken of mankind! (Personkind?)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TRUTH REARS ITS UGLY HEAD:

As Lincoln pointed out, you cannot fool all of the people all of the time. It has finally become embarrassingly evident that most of the pap emanating from the UNIX public relations departments is pure natural fertilizer. WE are seeing the beginnings of another Chick Little charge, this time in the reverse direction.

UNIX persons (opinion-leaders) are finally noticing, in print, that there is no such thing as a UNIX standard! UNIX/WORLD Executive Editor Philip J. Gill points out (issue #3, p. 7) that there are MANY UNIXs and that no two are compatible! UNIQUE dated August but arrived 20 September) September UNIX REVIEW's publisher's column (p. 4) discusses the difficulties of administering small UNIX installations of 2 to 4 users and insists that UNIX has lousy documentation. 3000 page manuals, but lousy documentation. September MINI-MICRO SYSTEMS' publisher's column (p. 11) flatly asserts that UNIX's time has passed. August DATAMATION magazine blasts UNIX. Even PC World's David Bunnell, bless him, has noticed the mismatch between AT&T/UNIX and the mass marketplace (Oct. p. 13).

(Let us quote from the last paragraph of MINI-MICRO's editorial: "UNIX brought the important of operating-system software into prominence. It has served its purpose. Let's now benefit from the experience and move on to the newer, more capable and fully integrated operating-system solutions...")

No, UNIX is not likely to lie down and die, at least not in the foreseeable future. But neither is UNIX going to expand (proportionally) much beyond the niche it now has.

As we pointed out way back in June '82, there is no such thing as "the best operating system." The concept of "best" is relative, in this case relative not only to the computer and the application but also to the user. It was absurd to assert that UNIX was or ever would be the best operating system for the dentists and candle-makers who comprise the mass personal computer marketplace. Finally, Felgercarb is no longer out of step, the UNIX gurus having changed step to match (without noticing him, to be sure — this is a tiny, obscure newsletter in an 'out' backwater, remember?).

If we were nicer than we are, we would not point out that we told you so.

AT$T PERFORMANCE:

Some comparative benchmarks have been run on some of AT&T's new UNIX machines. The first pitted the 3B2/300 against six other machines in the $12K to $20K class, and featured computations and RAM read/writes. The AT&T machine came in next to last, beating out the 8086-based Altos. The 68010-based Convergent Technologies Miniframe finished first. The AT&T machine came in in the middle in terminal support tests, and finished first in the disk performance test. (You don't suppose AT&T has learned through all the years that UNIX really DOES need a FAST hard disk, NOT just a hard disk?) The 8086-based Altos finished last in all three benchmarks. (Oct 1 EN, p. 21)

SYSTEM V & the 80286:

All of your probably know that SYSTEM V is the dialect of UNIX which is, no lie, gonna be the standard and conquer the world. That's because AT&T, the world's greatest and most experienced marketers, are behind it. (What do you mean, we have overspent our allotment of sarcasm?) Well, Motorola has had an AT&T- approved SYSTEM V release for some time now but the 80286-based release (called 'port' in the UNIX world) is seriously behind schedule — and may have some other problems related to the schedule. We thought you might like to know how this state of affairs came about:

The first thing you should now is that the SYSTEN V ports for the various microprocessors are being performed FOR AT&T, who will own the ports (code) when they are done, as they now own the 68000 port. Therefore, AT&T is also setting/has set the rules under which the ports are to be performed. Remember, AT&T wants ports which are COMPLETELY compatible with its own code.

The second thing you should know is that Intel, makers of the 80286, had already adopted Microsoft's XENIX, a UNIX clone, as it official internal operating system. So when Intel agreed to cooperate with AT&T in obtaining an 80286 SYSTEN V port, they naturally went to Microsoft. Microsoft read the contractual ground rules established at AT&T, which required that EACH LINE OF CODE changed had to be justified in writing and heavily documented. Like we said, AT&T wants ports which are COMPLETELY compatible with its own code.

Microsoft therefore learned quickly that it could not 'leverage' off the very substantial work it had done on XENIX, its UNIX clone. SO IT TURNED THE JOB DOWN!

Imagine the feeling Intel must have had to suddenly realize that their own internal standard operating system, XENIX, was going to be unrelated to the absolute, definite, for-sure future operating system standard, SYSTEM V! So, with heavy heart, Intel managed to persuade Digital Research (DRI) to take on the 80286-SYSTEM V port. (Some publications wrote at the time that DRI 'won' the port over Microsoft. That is not in accordance with the facts.)

Well, DRI is late delivering the 80286-SYSTEM V port and both AT&T and Intel are afraid that DRI is not going to

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

promote its SYSTEN V port even after (if?) it delivers. You do not suppose that DRI has detected which way the UNIX wind is blowing, do you? (The above information was abstracted from David Fiedler's UNIX newsletter Unique and from Oct. 1 EN, p. 21.)

UNIX/WORLD WATCH:

If you have a real interest in the high-end micro and/or UNIX world, you really should subscribe to this magazine. They have a special on now; 12 issues for $18. Call (415) 949-3737, CA time. Considering this is the usual commercial magazine with the usual advertising, it is remarkably if not TOTALLY candid.

For instance, Vol 1 #5 asserted editorially that UNIX was not EITHER a standard, and then published lots of material to prove that. Vol 1 #6, the current issue, editorially asserts that UNIX is not even transportable, and then they publish lots of material to prove that too.

Every magazine has a problem (aside from the necessities associated with those darn ads) and UNIX/WORLD's problem is its publisher, John Knapp. His techno-politics are somewhere to the right of Daddy Warbucks — in issue #5 he seriously proposes to stick a .45 handgun into the ear of all you personal computer users to force you back onto time-sharing systems with variable response times of up to 30 seconds. Knapp obviously believes that instant response and isolation from 'the computer is down!' isn't worth anything and in fact is a roadblock to true progress (what he means is a roadblock to the sales of his advertisers, who ALL make multi-user systems). He also obviously believes all that advertising crappola about how many users a given system will support — more on this real soon.

YOU WILL NOT BELIEVE THIS:

Beginning on p. 68 in issue #6 UNIX WORLD has a review of AT&T's 3B2/300 desktop uh, supermicro. This micro is based on a 7.5MHz WE32000 (nee BELLMAC 32). The WE32000 is an honest 32-bit micro which has, for instance, a 32-bit barrel shifter. In that later sense it is akin to the 68020. However, we have reported in the past that the WE32000 is not an especially hot performer and that in fact it is slower than a 68000 (and the Z8000, and the 80286, and...).

THE PROGRAMMER'S SHOP™

helps compare, evaluate, find products. Straight answers for serious programmers.

SERVICES
- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature free
- BULLETIN BOARD - 7 PM to 7 AM 617-826-4086
- Dealer's Inquire
- Newsletter
- Rush Order
- Over 500 products

"Guaranteeing" Software – Possible?
It is a very messy issue. Trust is required; but trust means that the publisher, retailer or user gets left open to losing, being hurt. The Programmer's Shop trusts customers and **guarantees SOLID VALUE.** Ask about it. The next "*Programmer's Letter*" offers ideas.

RECENT DISCOVERIES
ProYAM Communications Package - everything a programmer would want. TTY, VT 100, 3101, MODEM 7, BBS, Remote, macros, windows. MSDOS. $139.

"C" LANGUAGE

	OUR PRICE
MSDOS: C86-8087, reliable	call
Instant C - Inter., fast, full	500
Lattice 2.1 - improved	call
Microsoft C 2.x	329
Williams, debugger, fast	call
CPM80: Ecosoft C-now solid, full	225
BDS C - solid value	125
MACINTOSH: Softworks	365
Megamax-object, full	295
Consulair's MAC C	295

Compare, evaluate, consider other Cs

BASIC

	RUNS ON	
Active Trace-debug	86/80	75
BASCOM-86 - MicroSoft	8086	279
BASIC Dev't System	PCDOS	115
BetterBASIC - 640K	PCDOS	185
CB-86 - DRI	CPM86	439
Prof. BASIC Compiler	PCDOS	89
Databurst - screens	MSDOS	215
SCREEN SCULPTOR	PCDOS	119

Ask about ISAM, other addons for BASIC

SERVICE
ADVICE - We help choose a product to fit your work. Even if you don't know of a product, we'll help.

EDITORS Programming

	RUNS ON	OUR PRICE
BRIEF - Intuitive, flexible	PCDOS	195
C Screen with source	86/80	75
Epsilon - like EMACS	PCDOS	195
FINAL WORD-for manuals	86/80	215
MINCE-like EMACS	PC/80	149
PMATE-powerful	8086	195
VEDIT-full, liked	86/80	119

UNIX PC

COHERENT - for "C" users	PClike	475
COHERENT-NCI-Realtime	PClike	call
XENIX - plus C to MSDOS	PC	1295

Ask about run-times, applications, DOS compatibility, other alternatives. UNIX is a trademark of Bell Labs

LANGUAGE LIBRARIES

GRAPHICS: GraphiC-source in C	MSDOS	195
GRAPHMATIC-3D, FTN, PAS	PCDOS	125
HALO-fast, full-all lang.	PCDOS	175
FILE MGNT: BTrieve-all lang.	MSDOS	215
CIndex + -source, no royal.	86/80	375
CTree-source, no royal.	ALL	375
dbC ISAM by Lattice	8086	250
db VISTA-"Network" Structure	MSDOS	465
PHACT-up under UNIX, addons	MSDOS	250
OTHER: CUtil by Essential	MSDOS	139
Greenleaf - 200 +	MSDOS	165
CSharp - **Real-Time**	MSDOS	600
PORTABLE C to PC, Mac, II	Many	125
SOFT Horizons - Blocks I	PCDOS	139
SCREEN: CURSES by Lattice	PCDOS	125
CView - input, validate	PCDOS	195
MetaWINDOW - icons, clip	PCDOS	139
PANEL - many lang, term	MSDOS	265

FORTRAN

	RUNS ON	OUR PRICE
MS FORTRAN-86 - impr.	MSDOS	$ 239
DR Fortran-86 - full '77'	8086	280
PolyFORTRAN-XREF, Xtract	PCDOS	165

OTHER PRODUCTS

Assembler & Tools - DRI	8086	159
Atron Debugger for Lattice	PCDOS	395
cEnglish - dBase C	MSDOS	795
C Helper: DIFF, xref, more	86/80	135
CODESMITH-86 - debug	PCDOS	139
MacASM-full, fast, tools	MAC	100
MBP Cobol-86 - fast	8086	695
METAWINDOW - graph, fonts, clip	PCDOS	135
Micro: SubMATH-FORTRAN full	86/80	245
Microsoft MASM-86	MSDOS	85
MSD Debugger	PCDOS	119
Multilink - Multitasking	PCDOS	265
PC/FORTH + -well liked	MSDOS	219
PFIX-86 Debugger	MSDOS	169
PL/1-86	8086	495
Polylibrarian - thorough	MSDOS	95
PolyMAKE	PCDOS	95
PROFILER - flexible	MSDOS	125
Prolog-86-Learn, Experiment	MSDOS	125
SLK/F - Copy Protection	PCDOS	150
SYMD debugger-symbols	PCDOS	119
TRACE86 debugger ASM	MSDOS	115

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and **POs**. All formats available.

Call for a catalog, literature, and *solid value*

800-421-8006

THE PROGRAMMER'S SHOP™
128 J Rockland Street, Hanover, MA 02339
Visa Mass: 800-442-8070 or 617-826-7531 MasterCard

8517

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

We are about to talk about horrendously slow WE32000s, but darn it, it does not (necessarily) support our assertions about the relative speeds of those micros because of BIG BIG problems elsewhere!

The next thing you should know is that the performance data we are going to be writing about has been confirmed by AT&T: "After running the benchmarks, I just did not believe the numbers. First, the machine was much too slow; second, the single precision was slower than double precision. I was sure that there must be a bug somewhere or that I was doing something wrong. I called Ram Chelluri (a benchmark expert) at AT&T Technology and confirmed the numbers."

Here are the numbers that even the guy doing the benchmarking did not at first believe:

	32-BIT	64-BIT
ADD	3922	2128
MULT	9346	7463
DIV	5319	3663

Those times are in microseconds! Gentlepersons, the numbers above are IMPOSSIBLE, even though we ARE talking about an IEEE- compatible implementation. That is, they are impossible if you make the assumption that the folks who implemented that floating point package on the WE32000 knew what they were doing. There are AT LEAST three readers amongst the 700+ DTACK subscribers who can confirm what we are about to tell you (right Chet, David, Ulrich?):

Before we go on, keep in mind we are talking about a brand-new $16,000 supermicro — it says supermicro right here in the promotional literature — and IT IS 5 TIMES SLOWER THAN A $69.97 VIC-20!

CRASH AND BURN:

The idiots at AT&T obviously wrote that (IEEE compatible) floating point package in a high level language! Yes, 'C': you didn't think AT&T would use LISP or ADA, did you? How can we tell? The clues are obvious to anybody who knows anything at all about floating point packages:

For starters, divide is nearly twice as fat as multiply. That can't happen in assembly language; in assembly language matters are reversed and multiply is always a lot faster than divide. What we have here is a case of either a poor algorithm being selected for multiply or, more likely, the HLL algorithm selected for multiply being a very poor match for how the compiler works!

It should not be surprising that single precision is slower than double precision; this is the sort of B.S. that big organizations foist on us dummies all the time (in the past it was to save bytes). What we obviously have here is conversion to double precision, the operation performed, and then conversion back to single precision. The ridiculously longer time required is a function of the fact that the operation is being done in the high-level language 'C'.

Remember, we are talking about the fundamental software support provided by AT&T for its 32-bit $16,000 "supermicro." And it is five times slower than a VIC-20, we kid you not! (The author of the review also benchmarked a slow 8MHz 68000-based computer, the Scorpion, and it ran 19 times faster while using an IEEE format F.P. package.) So what does the author conclude at the end of the review? THE FOLLOWING IS A CONDENSATION OF THE LAST PARAGRAPH OF THE REVIEW. NO NEW WORDS HAVE BEEN ADDED; NO WORDS HAVE BEEN CHANGED.

"If IBM or Digital Equipment had produced it, then I probably would have given it a rave review. When you judge the 3B2/300 purely on its price/performance, it is an excellent machine. In fact, I want one, and I will suggest it to my clients."

RAVE REVIEW? PRICE/PERFORMANCE? 5 TIMES SLOWER THAN A VIC-20? The old geezer over there with the lamp, the one who is frantically trying to buy a Concorde ticket to the next continent, is Diogenes.

OPT-TECH SORT™

SORT/MERGE program for IBM-PC & XT

Now also sorts dBASE II files!

- Written in assembly language for **high performance**
  Example: 4,000 records of 128 bytes sorted to give key & pointer file in 30 seconds. **COMPARE!**
- Sort ascending or descending on up to nine fields
- Ten input files may be sorted or merged at one time
- Handles variable and fixed length records
- Supports all common data types
- Filesize limited only by your disk space
- Dynamically allocates memory and work files
- Output file can be full records, keys or pointers
- Can be run from keyboard or as a batch command
- Can be called as a subroutine to many languages
- Easy to use, includes on-line help feature
- Full documentation — sized like your PC manuals
- **$99** —VISA, M/C, Check, Money Order, COD, or PO
  Quantity discounts and OEM licensing available

To order or to receive additional information write or call:

OPT-TECH DATA PROCESSING

P.O. Box 2167 Humble, Texas 77347

(713) 454-7428

Requires DOS, 64K and One Disk Drive

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LESSONS LEARNED:

If you have been paying attention, you will know that we have been espousing the idea of writing important stuff like floating point packages and operating systems and high level languages in assembly for the past 3 1/2 years. (Longer than that, really, but what happened before we started putting our ideas in print doesn't count.)

On the other hand, nearly 900 of our 700+ subscribers have written to your stupid FNE patiently explaining to him what a wonderful idea 'transportable' high-level languages are. They have even tried to explain the wonderful 80%/20% (or was it 90%/10% ?) rule to him. But your FNE, in his ignorance, has resisted all these wonderful ideas. Possibly because he does not like 32-bit $16,000 supermicros which run 5 times slower than a VIC-20? (You know, at the moment we don't feel the least bit ashamed of the obvious sarcasm in this paragraph. Oh yes: 900 out of 700+ courtesy of the photocopy machines.)

There is more: the same review, on p. 71, knocks the theory of the overhead-less multitasking operating system in the kazinga. Wall clock time runs about 60% ahead of CPU time on a matrix test which could be benchmarked with an hour-glass — provided you had an hour-glass that lasted two hours! Oh, yes: where is the fabled 'C' compiler with only a 2-1 HLL overhead? Out for a beer with the Loch Ness monster and the Abominable Snowman?

This $16,000 superturkey requires 98 minutes to invert a 50 x 50 matrix. For big matrices, the inversion time is determined by the time required to perform N cubed multiplies and adds. The HALGOL 64-bit floating point package did a multiply and an add in about 150 microseconds (issue #17 p. 20). That was before we sped up the multiply but we should also make allowances for pivoting. We're guessing 160 microseconds, including pivoting, per operation pair. For a 50 X 50 matrix that's 125,000 times 160 microseconds or 20 seconds, that IS 294 TIMES FASTER THAN AT&T's $16,000 SUPERTURKEY!

LET US REPEAT THAT:

The HALGOL 64-bit floating point package can invert a 50-rank matrix about 300 times faster than the $16,000 AT&T 3B2/300 as measured by wall clock time. If we measure CPU time, which is only of interest to accountants and advertising copy writers, HALGOL is STILL 180 times faster. In making these comparisons you will remember that a Concorde supersonic jet transport cannot fly 100 times faster than a normally athletic 14 year-old girl can run.

Others are beginning to notice.

In David Fiedler's most recent issue of UNIQUE, his UNIX newsletter, he asserts: "...talks we've had with other 3B2 owners and our own tests lead us to believe that AT&T has oversold the 3B2's power by several hundred percent. Your editor even was present at a special executive briefing by Ian Ross, the President of Bell Labs, during which Ross admitted that the 3B2 could serve no more than 5 or 6 users. We hope and expect that AT&T will be more reasonable with its future claims..."

AT&T had previously claimed that one 3B2 could support the combined populations of China, India, Japan and Outer Mongolia. In fact, if any one user tried to calculate a logarithm, for instance, everything is gonna come to a real quick stop with even 5 or 6 users!

Yes, we know it is slightly unfair to compare HALGOL's non-IEEE double-precision floating point execution times to the floating point times of the 3B2 superturkey, which ARE IEEE compatible. But that should only count for about 8 or 10 to 1 in speed. And besides, we could have compared against Chet's compiled BASIC or Ulrich's Inter68 PASCAL packages, both of which support the Nat Semi math chip and so are about 1,000 times faster than AT&T's $16,000 superturkey!

ALTERNATE ARCHITECTURES:

Risk, er, RISC? The Nov issue of Byte has a very good overview of the RISC, or Reduced Instruction Set Computer. RISC's proponents assert that a RISC can provide better performance than existing CISCs, or Complex Instruction Set Computers. Since John Markov did a excellent job of presenting the case FOR the RISC in his Byte article, let us concentrate briefly on an opposing view.

Did you know that the 8080 can outperform the 4004? That the 8086 can outperform the 8080? That the 68020 can outperform the 8086? Oh? You knew all that? Of course you did! No, we are not trying to insult you; we are trying to make the point that the 4004 should be compared to the Autonetics PPS-4, the 808 to the 6800, the 8086 to the Z80000 and the 68020 to the Z80000 or the 80386. In other words, the performance of microprocessors should be compared to other contemporary microprocessors.

Therefore, we are troubled when some academics, however well-meaning, compare a brand-new first or second-silicon prototype with 1979 technology (an early silicon 8MHz 68000 with 2 wait states) and proclaim the superiority of that prototype. There is a lot of that going on in the RISC camp. Oh, yes: we forgot to mention that the opposite of the RISC, the CISC, is typified by the 68000 and the 68020. The RISC proponents seem to be asserting that they are going to kick the stuffings out of the 220,000-transistor 68020 with a RISC having fewer than 50,000 transistors!

We are not completely opposed to the arguments of the RISC crowd. In particular, WE GREATLY FAVOR LOTS OF ON-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The C Interpreter: *Instant-C*™

Programming in C has never been Faster. Learning C will never be Easier.

Instant-C™ is an optimizing interpreter for the C language that can make programming in C three or more times faster than using old-fashioned compilers and loaders. The interpreter environment makes C as easy to use and learn as Basic. Yet ***Instant-C***™ is 20 to 50 times faster than interpreted Basic. This new interactive development environment gives you:

Instant Editing. The full-screen editor is built into ***Instant-C***™ for immediate use. You don't wait for a separate editor program to start up.

Instant Error Correction. You can check syntax in the editor. Each error message is displayed on the screen with the cursor set to the trouble spot, ready for your correction. Errors are reported clearly, by the editor, and only one at a time.

Instant Execution. ***Instant-C***™ uses no assembler or loader. You can execute your program as soon as you finish editing.

Instant Testing. You can immediately execute any C statement or function, set variables, or evaluate expressions. Your results are displayed automatically.

Instant Symbolic Debugging. Watch execution by single statement stepping. Debugging features are built-in; you don't need to recompile or reload using special options.

Instant Loading. Directly generates .EXE or .CMD files at your request to create stand-alone versions of your programs.

Instant Floating Point. Uses 8087* co-processor if present.

Instant Compatibility. Follows K & R standards. Comprehensive standard library provided, with source code.

Instant Satisfaction. Get more done, faster, with better results. ***Instant-C***™ is available now, and works under PC-DOS, MS-DOS*, and CP/M-86*. Money back guarantee within 30 days.

Find out how ***Instant-C***™ is changing the way that programming is done. ***Instant-C***™ is $495. Call or write for more information.

Rational Systems, Inc.

(617) 653-6194
P.O. Box 480
Natick, Mass. 01760

Trademarks: MS-DOS (Microsoft Corp.), 8087 (Intel Corp.), CP/M-86 (Digital Research, Inc.), Instant-C (Rational Systems, Inc.)

Moving?

Please send us both your old and new addresses, with Zip Codes. If possible, include the address label from a recent issue of *Programmer's Journal*. Allow three to four weeks for processing.

Programmer's Journal
P.O. Box 3000
Dept. EE
Denville, NJ 07834

BOARD REGISTERS, especially when used in overlapping banks as the RISC machines do. If the RISC boys had had their act together (along with some commercial funding) back in the 1976-8 time frame when 50,000 transistors on a chip was the state of the art, they might well have been able to prove their point, at least for a while.

Today we think the RISC is doomed before it ever sees commercial reality (as a one-chip microprocessor). What Motorola's designers have done with that huge number of transistor-equivalents on the 68020 is create an architecture with multiple execution units acting in parallel under the direction of that complex microprogram which the RISC folk decry. A RISC machine (as we understand it) inherently has only a single execution unit on-chip. Since coordinating multiple execution units is inherently a complex task, it is our opinion that time has passed the RISC concept by.

(We hope it is obvious that a chip with effectively-coordinated multiple execution units is always going to beat the heck out of a chip with only a single execution unit.)

Meanwhile, the folks at Stanford do not want to play sycophantic second-fiddle to the Berkeley RISC folk and so have decided to conquer the world with a compiler. A very, very sophisticated compiler. (Be sure to read Markov's Byte article all the way to the end.) How many of you have noticed that the success or failure of the transputer will depend more on the success of OCCAM, which is a data-flow compiler, than on the transputer itself? True, they have to get the silicon to work first but that silicon won't be worth a tinker's dam if OCCAM cannot correctly coordinate multiple execution units in a general-purpose environment.

Best of all, ALL THIS STUFF HAD EITHER BEEN BUILT OR IS GETTING BUILT. For real! These are exciting times! If you are churlish enough to disagree, go watch a Gilligan's Island rerun and let the rest of us enjoy ourselves.

Subscriptions to DTACK GROUNDED are $15 for 10 issues in the U.S. and Canada (U.S. funds), or $25 for 10 issues elsewhere. Make the check payable to DTACK GROUNDED. The address is:

DTACK GROUNDED
1415 E. MCFADDEN, STE. F
SANTA ANA, CA 92705
(714) 835-4884 □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

REDIRECTIONS

by Frank D. Greco, AGS Computers, Inc.

ARTIFICIAL INTELLIGENCE

There has been a recent rash of products and plans for software dealing with machine or artificial intelligence. Most people believe that artificial intelligence ("AI") is the next major step in software innovation. Some, like the Japanese, are establishing massive projects based on AI applications. Symbolics, Quintus, Prologia, and Smart Systems are some of the new companies that hope to make it "bigtime" in the software arena. The Japanese Ministry of International Trade and Industry (MITI) has a fifth generation project underway in which researchers are attempting to create machines that have a version of the programming language Prolog (called "ML1") as their native machine language. The Department of Defense in our own country is also very interested in using AI-based software. Their purpose is obviously for military applications. Some other companies are using AI as the latest buzzword in the computer field (you know, something to talk about at the next tennis or golf match). They use the term AI as a simple attention-getter, and most of the time the usage is incorrect. These pseudo-AI systems are merely clever applications of existing programming techniques.

However, one true application of AI is something called an "expert system." There are some expert systems already available for the latest generation of microcomputers. An expert system is a software and/or hardware system that will simulate a human expert in a given field. Initially, a database of facts, and rules governing those facts, is created with the help of a human expert. In the AI field, this database is called a "knowledge base." The human expert either enters facts indirectly into the knowledge base through an interview with a "knowledge engineer" (Steve Jobs loves that MACphrase), or directly through an intelligent interface. This knowledge base commonly has the appearance of textual IF-THEN rules. Unlike a conventional database, a knowledge base stores facts and relationships. These facts and relationships can be used to derive other facts. This is done by another integral portion of an expert system called the "inference engine." The inference engine uses the information in the database to logically deduce an appropriate answer.

Expert-Ease from Jeffrey Perrone & Associates, Inc (415) 431 - 9562, is one of the first expert systems for the microcomputer world. It uses a spreadsheet-like user interface to input the rules (or knowledge base). Expert-Ease for the IBM PC is written in P-System Pascal. For large knowledge bases, I would guess that Expert-Ease would be terribly slow due to the inherent tree-searching algorithms involved in typical AI work and, of course, the P-System's interpretive nature.

AI programs typically build a tree representation of the knowledge domain and traverse the entire tree trying to find possible solutions. This is similar to the many chess-playing computer programs out there. Of course, tree traversal tends to be exponential; as the number of nodes increases, the solution space grows quickly. Many researchers in the academic community are currently mixed in their feelings as to what type of heuristics help in minimizing the tree searching and thus reducing the possible solution space. Now you know the reason why AT&T Bell Laboratories is allowing Unix creator Ken Thompson and chess colleague Joe Condon to continue to work on Belle, their chess hardware/software! Further work on chess programs will provide many insights into how to intelligently "see our way through the trees," so to speak.

COMPANIES INVESTING IN ARTIFICIAL INTELLIGENCE

Major corporations have budgeted large amounts of money toward the artificial intelligence area. At the last Logic Programming symposium in Atlantic City, New Jersey, I noticed companies like IBM, DEC, AT&T, TI, and a host of Japanese companies. Mr. Doug DeGroot, a researcher from IBM's Symbolic Computation Lab, was a key organizer. Most of the attendees were either LISP proponents or PROLOG backers.

Digital Equipment Corporation is extremely interested in the AI area. DEC has their own Artificial Intelligence Technology Center in Hudson, Massachusetts, employing over 100 researchers and engineers. They have already designed an AI system that helps their own marketing people to intelligently configure the hardware for a computing system. DEC is hoping to gain a quick lead in the commercial AI market; they are already marketing at least a half-dozen third-party AI systems. Some of these companies are Carnegie Group, Inc. (SRL+), Gold Hill Computers (LISP), Inference Corp. (Automatic Reasoning Tool), Prologia (Prolog II), and USC Information Sciences Institute (Interlisp). With their new, powerful and compatible 8600 series VAX, they seem well poised for the coming generation of smart software.

Not to be outdone, IBM has released an expert system called PRISM, PRototype Inference SysteM, written in ISO Pascal for the VM operating system running on mainframe computers. They have also announced the PSC Prolog language for mainframes. This version of Prolog interfaces with their SQL/VM (Structured Query Language on the VM OS) and also with the IBM CMS Executive Command language (REXX). IBM also introduced a software product, "Handy," which is a user

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

interface to AI systems. Handy is an IBM PC based system that has elements of windowing, color animation, speech synthesis, and other video-graphic goodies.

Xerox recently formed an Artificial Intelligence Systems Business Unit to create expert systems for the commercial marketplace. In response to a question concerning the possibility of expert systems replacing a human professional, Beau Sheil, Xerox manager in the AI group, said, "No one wants 95 percent of a doctor, but something that will make a doctor 5 percent more productive suddenly becomes very attractive."

Texas Instruments has a program called "Personal Consultant" that runs under MS-DOS. This is another expert system development tool. Similar to Expert-Ease, the Personal Consultant allows a knowledge engineer to input a set of rules (up to 400) that are consulted during the inference stage. It also has a regression testing facility that records answers during a consultation with the human expert and iteratively refines the knowledge bases.

This iterative refinement of a knowledge base reminds me of a computer game that I played while an undergraduate; that was called "animal." The object of the game was for the computer to guess what animal you were thinking of. If the program could not guess your animal, it asked you to enter the qualities of your animal that differentiated it from other animals. It added this information to its database and subsequently correctly guessed this animal in the future. Well, an expert system is a sort of second-generation "animal" game. The researchers at MIT and Carnegie Mellon are probably gagging on that statement!

The artificial intelligence area is not relegated to the big boys only. As reported previously in this column, Microsoft has their own AI group that is working on consumer products. Even Lotus has already spun off a small group of researchers to do AI work and is currently funding them.

Some companies have written program editors that utilize rudimentary artificial intelligence techniques. These editors are called language- or syntax-directed editors. The knowledge domain of these editors is the syntax of whatever language you are editing. They catch syntax errors on the fly. Some BASIC interpreters perform syntax checking when you enter a line I remember an old Tektronics 4051 that did this, but the checking was only performed in the universe of the current line. The interpreter did not know about the line above, and it did not have any information about what was expected in the next line. Language-directed editors, being more than simple-minded character-oriented editors — like IBM's Personal and Professional Editors for PC-DOS, Gosling's EMACS, and Joy's VI editors for Unix — know about what you are editing. This new type of editor will not allow you to make syntax errors. And it will warn you of some possible semantic errors.

APPLESAUCE

ExperTelligence, Inc., is currently developing an expert system for the Macintosh. They are also working on incremental compilers, like MacBASIC, for the LISP and Logo languages for the MAC.

Megamax, Inc. (214) 987-4931, has released a full C compiler for the MAC that has in-line assembly and full access to the Macintosh ROM. The package costs $299.95.

Rumor has it that Think Technologies, the company that wrote the MacPascal interpreter, is hard at work on a similar product for the C language on the MAC. Preliminary indications are that it will be similar in usage to the Pascal package, that is, only for academic usage, not really for professionals.

Other companies selling C compilers for the MAC, or at least announcing them, include Consulair Corp (415) 851-3849 C interpreter, Hippopotamus (408) 730-2601, Manx Software (201) 780-4004, Mark Williams (312) 472-6659, Megamax Inc. (214) 987-4931 and Softworks Ltd. (312) 327-7666.

Speaking of the C language, if you want any information about the ongoing ANSI standardization of the C programming language, write to Jim Brodie, X3J11 Chair, Motorola, Inc., 2900 South Diablo Way, Tempe, Arizona, 85282. The committee Brodie heads, dubbed X3J11, was scheduled to meet December 3-7, 1984, in Tempe. If these standards are like other standards, they will be too late and too rigid. By the time the committee has finalized a standard for the C language, Modula II or Prolog will be in high gear. By the way, IBM has a seat on the ANSI C Committee.

Apple finally acknowledged a design flaw in the IIc modem port. Apparently the handshaking timing is off and causes problems with non-Apple modems operating 8-bit communications at 1200 baud. If you can demonstrate the problem to your dealer, he is required to perform a system board swap free of charge.

Macintosh is NOT a trademark of Apple Computer! It is a trademark that is LICENSED to Apple. Currently both McIntosh, the audio company, and some bank that uses the MAC name are reportedly suing Apple for misuse of the name.

THE "YOU CAN LEAD A HORSE TO WATER BUT...WHY?" DEPARTMENT

IQ Software (817) 589-2000 has released CP/M for the Macintosh. Yes folks, get set for a load of software that does NOT

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

realize it is running on a machine with a bit-mapped screen, 4-channel sound, 500 remarkable ROM routines, and a mouse. That's one small step for (a) man, one giant leap backwards for computerkind!

Micro Focus, Inc. is ready to announce MAC COBOL for the Macintosh. Imagine seeing File Descriptor Sections, Data Divisions, and COBOL paragraphs displayed in a window on a MAC?! Ugh!

Remember we mentioned Unix on a MAC or Lisa in this column? Well, get set for a Super 68010 MAC with a 20 Meg hard disk running Berkeley 4.2 Unix and color graphics. Apple has been showing the beast to a few select OEMs. These OEMs have also reported seeing a color MAC with a 12-inch monitor. This is the FAT MAC; the recently announced 512K MAC was NOT the formal name for the FAT MAC.

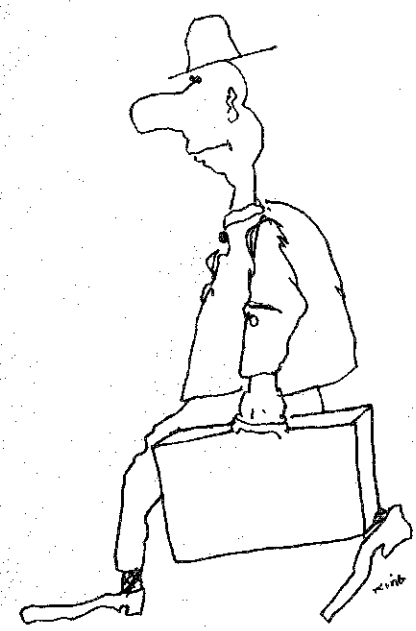
BLUE NEWS

SCI Systems, maker of the IBM PC's motherboard, has signed a letter of intent with IBM to supply some "guts" for the IBM PC AT. Other parts of the AT are made in Boca Raton. SCI has a box called the SCI 1000 that runs PC/IX.

Even though YE Data, Inc. from Japan makes the PC AT 1.2 Mbyte floppy drives, and Qume Corp. and Atlas Industries Ltd. are Big Blue's current favorites on the half-height drives, IBM will soon be manufacturing their own disk drives for their PCs. We can obviously expect IBM to make their own 1.6 Mbyte floppy drives for the AT; that is a huge market. But we also expect some super-fast voice coil drives when multi-user software arrives for the AT.

Many people look only at the data capacities when investigating hard disks; the access times of the disks should also be considered. Disk I/O bottlenecks are a large cause of system slowdown. They can make a fast processor appear to be slow. Multi-user operating systems require those fast voice coil drives instead of the cheaper and slower stepper motor drives. The current XT uses a stepper motor winchester with an average access time of 85 milliseconds; the PC AT also has a stepper motor winnie with a respectable 40 ms average access time. On the other hand, voice coil drives have access times in the neighborhood of 20 ms. Those developers thirsting for disk hog Xenix on the AT will look longingly at these new drives.

I would bet that IBM will be showing 5 1/4 inch half-height winchester drives that have capacities of 40 Mbyte and larger



Famous Computer Personalities #504
Wadsley Bifwaite, inventor of the world's weakest operating system, PUNIX.

ADVANCED PROCESSORS ARE OK
TUNED SOFTWARE IS BETTER

Performance. Every user wants more of it.

Advanced hardware is one solution. But advanced hardware is difficult to obtain and expensive. Meanwhile, there are MILLIONS of existing machines waiting for a new generation of faster, more robust software. How will your software measure up?

Performance optimization should be considered part of the software development cycle.

Many programs spend 90% of the time executing less than 10% of the code. A small amount of time invested in optimizing these busy areas can yield big performance dividends. But rarely is it obvious just what pieces of source code account for 90% of the processor time. You need a sieve that can quickly isolate the big pieces (timewise) of your program.

CODE SIFTER is the tool you need to identify the time consuming sections of your EXE or COM file.

It observes your running program and generates a statistical report that indicates which areas of your program are prime candidates for your optimization efforts. The statistical output includes the symbols from your LINK map and can be directed to your display, printer, or to a file.

CODE SIFTER has advanced features.

Like a user friendly interface with function keys and online help. An adjustable sampling rate provides more accurate statistical data. An iteration option can rerun your program a number of times to reduce the busy areas to very narrow ranges. A demonstration program (with source) is provided for use with the step by step tutorial in the manual so that you can become productive in just a few minutes. CODE SIFTER works with any source language and does NOT require a knowledge of assembler to use. And if you are still not convinced – we will loan you a demonstration version.

David Smith Software
Box 25A R.D.#3
Oxford, N.Y. 13830
(607) 843–6209

$119

Requires
IBM–PC or XT
128K DOS 2.x

IBM is a registered trademark of International Business Machines Corporation

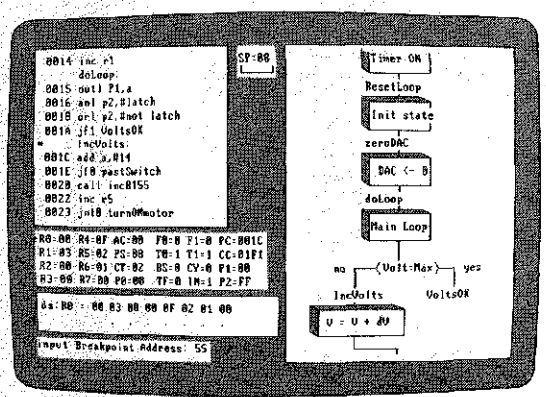
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

for the PC AT. And by putting several platters in the same box with some additional read/write heads, IBM can offer capacities in the 200 Mbyte range. You have to remember that IBM has mastered thin-film technology and manufacturing. They could use this leverage to build ultra-high density drives. Also, the replacement for the PC family is purportedly using 3 1/2 inch drives a la Apple Macintosh. Seagate is now even showing some OEMs a 20 Mbyte 3 1/2 inch winchester drive, and so are some Japanese manufacturers. IBM surely sees a very large market for them.

Now that the Japanese have announced erasable optical disks with capacities in the gigabytes, IBM is carefully watching and has several researchers looking into the matter. In Boulder, Colorado, IBM is working on a 5 1/4 inch optical disk for use with the PC. A 12 inch disk is being developed in Tucson, Arizona, presumably for the bigger machines. Both groups report to IBM in Rochester, New York, where there is a team doing technological development common to both. Optical media is being developed at IBM in San Jose, California.

IBM is also watching North American Philips which has developed a Compact Disk player with a standard disk interface. The read-only CDs, which are to be made by 3M, have capacities of 600 Megabytes.

Simulator/Debugger for IBM-PC for 8048 - 8051 - 7000 µCs



Execute and debug code for popular single chip microcomputers on your IBM PC. Dynamic display with windows for source code, control flow, registers, flags, memory, commands, and more! Set breakpoints, traps, etc. Cross-assembler & EPROM programmers, too.

	8048	8051	7000	Z8	
X-Assembler	✓	✓	✓	✓	$195
EPROM Prog	✓	✓	✓	✓	$245
Simulator	✓	✓	✓		$395

Demo disk and manual for simulator only $39.50

Cybernetic Micro Systems
Box 3000 ● San Gregorio, California 94074 U.S.A.
(415) 726-3000 ● Telex 171-135 Attn: Cybernetics

Someone that IBM should REALLY watch for CD units is Sanyo. At the Japan Audio Fair, Sanyo announced Laser-ERASABLE CD audio media. Right now, this Compact Disk unit is just for audio. However, once they solved the non-erasability problem for sound, computer usages are just around the corner. After all, the new Sony consumer CD decks have RS-422 interfaces!

If IBM is not so interested in the controversial MSX operating system from Microsoft, why did they spend $5000 for some MSX hardware? The New York based Qest Publishing, a MSX promoter, recently received the check from Big Blue for some Sony, Toshiba, and other Japanese made hardware/software.

NEXT GENERATION OF IBM PC'S — PC II

A very logical guess would have IBM using iAPX-186 microprocessors in the PC II, since these chips reduce the ancillary chip count (many support chips are on the 186 itself), thus lowering manufacturing costs. The 8086 is old hat already, and the 286 might not lend itself to cheap manufacture. Besides, the 286 has compatibility problems with 8088 software. I'm sure IBM does not want to contend with that can of worms.

Reportedly, Verbatim Corp. has a contract to supply IBM with 500,000 3 1/2 inch floppy disks a month. One would guess that they will be used in the upcoming PC II. And Toshiba is believed to be set to deliver 200,000 3 1/2 inch disk drives per month to IBM.

PHONE COMPANY CONTINUES TO LEARN or AS THE BELL TOILS

AT&T announced the 32100 microprocessor, an enhancement to the Bellmac-32A 32-bit microprocessor, at COMPCON in San Francisco. The 32100 is now used in the 3B2 and 3B5 computers as the main CPU, and in the 3B20S as an ancillary chip. The enhancements include 256 bytes of on-board cache memory, an I/O controller for multi- and co-processors systems, a 132-pin ceramic pin-grid array package, and unmultiplexed address and data ports. AT&T is using proven low-power CMOS technology in manufacturing the 32100; the chip dissipates between 0.5 and 1.9 watts. This new chip has about 180,000 transistors and runs at about 14 MHz. AT&T Bell Labs has said that this new chip is about 3 times faster than the old 32000 (Bellmac 32A) chip and 11 times faster than the original Bellmac 32. Even though AT&T has formally released the chip's specifications, they have no plans for selling it to third parties. Grumblers on the Unix network Usenet, are "flamed" (techie talk for being mad) that the coprocessor I/O controller has bugs. Thus there isn't a floating point processor available for the 3B2 and 3B5 series. Recent benchmarks in UNIX/WORLD magazine, UNIQUE

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

newsletter, Yates Ventures and on the "Net" (Usenet) have given the 3B2 microcomputer very low marks. The UNIX/WORLD review article says that the 8/16-bit IBM PC is 6 times faster for floating point multiply operations than the full 32-bit 3B2.

An interesting bit of advertising for AT&T Teletype's 5620 bit-mapped terminal appeared in a recent issue of ELECTRONIC NEWS. A photo of the 5620 was overlaid on a photo of the skyline of a major city. What was interesting was that right next to the Teletype terminal in subliminal fashion was the word "PLAYBOY" (the Playboy building is in the picture). Talk about suggestions to the subconscious.

DON'T BURY ME YET

Amidst the recent flurry of attention focused on the Unix operating system, one tends to forget Unix's ancestor MULTICS. Surprise, surprise, MULTICS still lives! Honeywell, who owns the rights to MULTICS now, has announced an update to the forgotten operating system. MULTICS Release 10.2 includes new data base features, multi-directional scrolling on any supported terminal, MULTICS FORTRAN that allows arrays as large as 64 Mbytes, and new communications software. MULTICS was a project funded by DARPA, the Department of Advanced Research Projects Agency, which had General Electric, MIT, and Bell Labs all involved. MULTICS was a project in the 1960s that attempted to develop a very powerful multi-tasking and multi-user time-sharing system. As far as I know, MIT was writing the user interface and Bell Labs was writing the file I/O routines (hence the elegant albeit slow Unix file system). GE's computer group was subsequently sold to Honeywell, who now markets MULTICS. MULTICS is still being used at MIT. If you call the MIT computer department and ask for the MULTICS administrator, you will find one MULTICS zealot. Take my word for it!

Another shot in the arm for the 6502 family came from Western Design Center (WDS) in Mesa, Arizona. Everyone knows that WDS developed a 16-bit version of the 6502, called the 65816, last year. But what everyone does not know is that WDS has licensed the new chip, which Steve Wozniak is reportedly tinkering around with, to Hyundai Electronics America. Hyundai is to build CMOS versions! Another licensee, GTE, is currently building the "65C816" for, presumably, OEMs that have an interest in providing lap-sized Apple II, Atari, or Commodore 64-compatible lappers.

AN EYE-OPENING LOOK AT MULTI-TASKING MS-DOS

For those of you who are waiting for a TRUE multi-tasking version of MS-DOS instead of the recent rash of products that offer "co- resident" programs, I'm afraid that you will probably have to wait a little longer. Some insiders have said that the serious work has just begun on making MS-DOS do more than one task at a time. Due to some serious internal limitations of the current DOS, which prevent it from becoming multi-tasking, Microsoft is considering rewriting DOS in mostly C language (for portability) from scratch. For those who do not know, MS-DOS is written in 8086 assembler.

One of the biggest restrictions on making DOS multi-tasking is the fact that each program CAN have its own stack segment. This makes it difficult for the processor to control the time-sharing between processes. Currently, MS-DOS cannot possibly know which process to return the control to if each one has its own stack. Having each program controlling its own stack also prevents MS- DOS from having "sharable" programs, that is, programs or procedures that can share one physical copy of an executable image in memory. This way, more than one user can share the instruction space of a running program and save on available memory for data. Those of you who have played around with the PC's INT 4B EXEC call know that another physical (!) copy of COMMAND.COM has to be loaded before doing the EXEC. Talk about kludges! Kind of reminds you of the MS-DOS inelegant implementation of pipes: temporary disk files operating sequentially. Unix runs pipes as parallel processes communicating in RAM.

NEW!
Advanced
Trace86™
Symbolic Debugger & Assembler Combo
• Full-screen trace with single stepping; Even backstepping!
• Write & Edit COM & EXE programs
• Conditional breakpoints (programmable)
• Switch between trace and output screen; Or set up two monitors
• 8087, 80186, 80286, 80287 support
• Write labels & comments on code
• Polish hex/decimal calculator
• and more . . . Priced at $175.00
To order or request more information contact:
Morgan Computing Co., Inc.
P.O. Box 112730, Dallas, TX 75011
(214) 739-5895

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Another multi-tasking MS-DOS roadblock is with the 8086/8088 itself. Neither the 8086 nor its baby brother the 8088 can be put into what is called "supervisory" mode; they are always in "user" mode. When a processor is in supervisor mode, user programs cannot call the lowest level routines explicitly. BASIC programmers cannot POKE bytes wherever they want in low memory. Assembler jocks cannot stuff characters DIRECTLY into the keyboard buffer. C sharpshooters cannot have pointers directly aimed at kernel address space. In a supervisory mode, all must go through calls in the nucleus or kernel of the operating system. The kernel obviously checks to see if the permissions are appropriate for the action requested. In this way, runaway programs (and programmers) cannot possibly disturb other programs in memory. The 68000 and Z8000 are microprocessors that can be put into supervisory mode and user mode.

Still another problem lies with the I/O routines in the PC's BIOS routines. Typically, when an I/O request is made to the BIOS, it is continued until it is completed. Check the disk I/O routines if you are not convinced. In a multi-tasking operating 0system, when a running program (or process) starts to perform some slow I/O, instead of waiting, the processor is switched to another process. When the OS comes back around to the original process after servicing others, the I/O continues where it left off. The PC's BIOS cannot relinquish its control once it has started. This is one reason why multi-tasking operating systems like PC/IX, Venix, and Xenix (they're all adaptations of AT&T Unix) do not use the ROM BIOS at all. All the low-level Unix routines are in the RAM-based kernel. The lack of memory management on the PC and PC XT is still another obstacle that has already been tackled in this column.

But along comes the IBM PC AT. The 80286 (or more formally the iAPX 286) can either be put into a supervisory mode for multi-tasking OSs like Xenix, or it can be in a user mode for the single-tasking PC-DOS. There is also a new version of the BIOS in the AT. Unix will still not use it, however, because of some philosophical design differences between it and MS-DOS. The AT's hard disk has an average access time of 40 milliseconds (about twice as fast as the XT's), which is respectable enough. And to top it off, the 286 has memory protection built right into the chip.

Now, if we have the concept of a PORTABLE (i.e., 8086, 68000, Z8000, Bell 32100, NatSemi 32000), C language based, truly multi-tasking MS-DOS with a more elegant file system (the current implementation is abysmal) and an improved, highly programmable shell instead of COMMAND.COM (I am starting to compare COMMAND.COM to EDLIN.COM, it's that bad) we have an interesting scenario unfolding in late 1985. Let us say that Microsoft calls it multi-tasking MS-DOS, "5.0" ("five dot oh") as a marketing dig at AT&T's Unix System V. The possibilities are very interesting.

	Microsoft's MS-DOS 5.0	AT&T's Unix System V
Multi-tasking	Yes	Yes
Multi-user	No, for a reason	Yes
Response Time	Depends on User	Depends on Everyone
Portable	Yes	Yes
Written in C	Yes, mostly	Yes, mostly
LANs	Yes, from MS-DOS 3.1	Sort of
Compatible Sftwr	Yes, from PC market	Sort of
Windowing Sftwr	Yes, integral Windows	Sort of
Mice	Yes	Sort of, non-portable
User Shells	Yes, visual shell avail.	Yes, many avail.
Disk Bound	Not really	Yes, very
Graphics	Yes, GKS compat.	Yes, NAPLPS compat.
Reentrant Code	Yes	Yes
Elegant I/O	Yes	Yes, for its time
RAM Disks	Yes, standard driver	Sort of, non-portable
Age	4-5 years	16 years
System Software	Yes, modern tools	Yes, but archaic
Support	Yes, help from OEMs	Yes, the Bell touch
Documentation	Yes, 1985's manuals	Yes, 1970's memos
Word Processing	Yes, bit-mapped Word	Yes, 3rd party
Text Processing	see Word Processing	Yes, archaic nroff
Current Std Vers.	1.0	5.3, 4.2, V7, Sys 3
Source Available	No	Yes, $43,000
Lineage	1970/1980's Unix	1960's MULTICS
Bug Fixes	"Never fixed any before"	"No bugs exist"

It becomes interesting, doesn't it. My last sardonic quip on bug fixes aside, Microsoft has several theoretical advantages. The greatest probably lies in the multi-user area. The boys from Bellevue believe that every user should get his/her own CPU and that these CPUs should all be networked. They would all share each other's local resources, like laser printers and hard disks. "Why timeshare a $20 microprocessor?" is their cry. Users want instant response times, not response times that are contingent on their neighbors' current work. And by learning concepts from Unix and improving them

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

dramatically, Microsoft is in a potentially very powerful position to control the small-machine operating system/environment area. The only realistic hurdles that Gates and company have to clear are AT&T's presence in the telecommunication/corporate world and IBM's eternal omnipresence. Big Blue has been relatively quiet in the operating systems area; too quiet. TopView is not the last word from the blue suiters, you can bet on that.

PINK NOISE DEPARTMENT

As reported in Redirections, start-up Advanced Matrix Technology, Inc.'s new better-than-letter-quality printer was too good to be true. As I surmised, since the founders were from the old Malibu printer company, which surprisingly folded (their printers were great for their time), AMT was walking a tightrope with their new business partners. Well, the romance is over. Three of the founders have resigned due to "philosophical differences with the board."

The nonprofit Unix organization, /user/group, has finally proposed the standard for all Unix systems. It includes a standard kernel, a C language interface that is entirely portable from machine to machine, the ability to be run as a subset of another operating system, standardized local network operations, and some possible extensions (on a standard?) for real-time programming. Write to /user/group, 4655 Old Ironside Drive, Suite 200, Santa Clara, California 95050.

Japan is very interested in the Unix marketplace, and AT&T helped them establish a Japanese Unix System Advisory Committee consisting of NTT, Fujitsu, Hitachi, Mitsubishi, NEC, Oki, and Toshiba. I wonder, with these very large Japanese companies (read: large Japanese resources) obtaining Unix SOURCE licenses ($43,000) and the Japanese history of very clever copying, what kind of non-Unix system they would develop and subsequently market without any AT&T license fees. If you think that Japan is going to blindly accept System V as an international standard for Unix, you are probably of the same ilk as the people who are saying that the Japanese cannot sell computers in this country because they don't know how.

Still another C compiler for the IBM PC market. EP Software (415) 924-3560, has a $195 C compiler that is Unix Version 7 compatible (no enumerated types and struct copies to functions). It comes with a symbolic debugger.

IBM's attempt at the Esperanto of computer languages, PL/I, is available for Unix from Language Processors Inc. (LPI) in Waltham, Massachusetts, (617) 890-1155. They also have Pascal, COBOL, BASIC, C, and even RPG II (ugh!) for Unix. As a matter of fact, you can mix routines written in different languages in one run-time module very easily, the reason being that all of the LPI languages compile to what they term "polyglots." This is similar to the p-system approach, except that the polyglots are eventually compiled into machine code. Most of the Unix-purist language writers out there all subscribe to the portability issue, but the others typically compile all other languages to C language. For example, FORTRAN 77 on a Unix system is typically compiled to C language, then compiled to assembler, then assembled to machine code. C is the ultimate generic assembly language.

The new Radio Shack DT-100 $799 terminals are really the excellent Wyse Technology WY-75's in Shack's clothing. Altos Computers recently bought 30 percent of Wyse Technology. Apparently, the multi-user market looks very promising to Altos, who's planning to announce a 286 machine soon (who isn't!). David Jackson, chairman of Altos, has said that Altos is already doing preliminary studies of the soon-to-be-sampled, full 32-bit Intel iAPX 386.

Since Wyse is doing fairly well in the terminals market, they have announced the WY-1000 microcomputer. It uses MS-DOS as the software engine and the 80186 as the hardware engine. Most PC-based software that is not tied to the specs of the 4.77 MHz 8088, I was told, should run on this machine. This could be a PC II compatible machine, before the original PC II is available! (I don't consider the Tandy 2000 a PC compatible, even if it does use a 186.)

Texas Instruments has announced their Advanced Video Display Processor (AVDP) for primarily videotex applications. This chip offers NAPLPS compatibility (256 X 210 with 16 colors), block move for rapid redraws, machine-level polygon-fill (and you thought the software-oriented MACPaint was fast!), sound synthesis with three voices and noise generation, and video overlay capability. It also supports 120 complex sprites per screen in 512 colors. This processor can change color palettes during retrace! TI is getting a lot of television manufacturers interested in including this chip in all of the newest TV sets for the average consumer. If Japan's and Europe's TV sets have this stuff already, it seems America is next.

An interesting job advertisement was seen in an electronics trade rag recently. Olivetti is looking for a hardware development manager in Cupertino who should have in-depth knowledge of the 80286. The ad says that "this manager will oversee a newly established group of 10 engineers involved with one of our most recently developed microcomputer systems." Hmmmmmm...

AT&T GETS GRAPHIC

AT&T announced some new graphics devices in connection with their 6300 PC. They released a board that can digitize

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Advanced Screen Management made easy

Now a professional software tool from Creative Solutions.

WINDOWS FOR C™

More than a window display system, WINDOWS FOR C is a video tool kit for all screen management tasks.

- **Pop-up menus and help files**
- **Auto memory management**
- **Keyboard interpreter**
- **Word wrap**
- **Auto scroll**
- **Highlighting**
- **Color control**
- **Overlay and restore**
- **Plus a library of over 50 building block subroutines**

Designed for enhanced portability. Easy to learn, easy to use.

Once you've tried WINDOWS FOR C, you'll wonder how you ever managed without it.

Full support for IBM PC/XT/AT and compatibles, plus interfaces for non-IBM computers; Lattice C, CI-C86, Mark Wm. C, Aztec C, Microsoft C, DeSmet C (PC/MSDOS),

NEW Ver. 3.1
Enhanced portability.
Topview compatible.
WINDOWS FOR C $195
(specify compiler & version)
Demo disk and manual $ 30
(applies toward purchase)
Full source available.
No royalties.

Creative Solutions
21 Elm Ave., Box T4,
Richford, VT 05476
802-848-7738
Master Card & Visa Accepted
Shipping $2.50
VT residents add 4% tax.

1	2
3	4

Have we got your number?

Is Programmer's Journal reaching you at the correct address? Please notify us of any address change or correction.

Is it nearing time to renew? Your Programmer's Journal mailing label shows when your subscription expires. Renew early to avoid any interruption in your subscription service!

Programmer's Journal, P.O. Box 3000, Dept. EE, Denville, NJ 07834

and store color images from any video source. Imagine this jazzy enhancement to all those PC databases out there! Real estate brokers can show images of houses to potential buyers, placement services can show photos of their clients next to their credentials, Football scouts can send photos of interesting prospects back home via modem, and think of the possibilities when a digital image is fed back into a painting program!

AT&T also showed a video display adapter (VDA) that generates a high resolution 512 X 256 in 16 color and intensity levels, or it can operate in 512 X 256 in 256 colors out of a possible 32,768. While that does seem too impressive, let me add that that resolution was on an ordinary home television. Yes, this system transforms your boob tube into a continuous tone high-res monitor. Apparently, AT&T is not the first to demonstrate this innovative engineering feat. Amiga Computer, one of the most popular home computer companies that has yet to sell a computer, showed their product a while ago to select OEMs. This was before the litigation with Atari and Commodore. AT&T is trying to position their VDA with mass-market oriented OEMs.

Speaking of Amiga, Commodore has consolidated efforts to direct the future development and marketing of the 68000-based home computer in Los Gatos, California. A former manager of advanced Apple II development, Rick Geiger, was named to the post of General Manager of West Coast Development for Commodore. I really cannot wait for the Amiga machine, if all those incredible rumors are true about its being an improvement over the Mindset machine (and cheaper). Commodore has plans to announce an inexpensive 8088 machine in the first quarter of 1985 and their Z8000 Unix box (actually COHOS, another name for Coherent from Mark Williams) in the second quarter. Commodore also has serious plans for the lap computer market. Former Northern Telecom engineer Dave Baraff is working on a cost-effective and original flat panel display that is both LCD and some sort of thin-film transistor device.

The Bottom Line, (800) 881-9855, a mailorder house for PC hardware items, has just initiated the sale of very inexpensive software. IBM's Macro Assembler is listed for $69.00, Microsoft's FORTRAN compiler for $244.00, PC-Mouse and PC-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Paint together for $149.00. They are from Milford, New Hampshire.

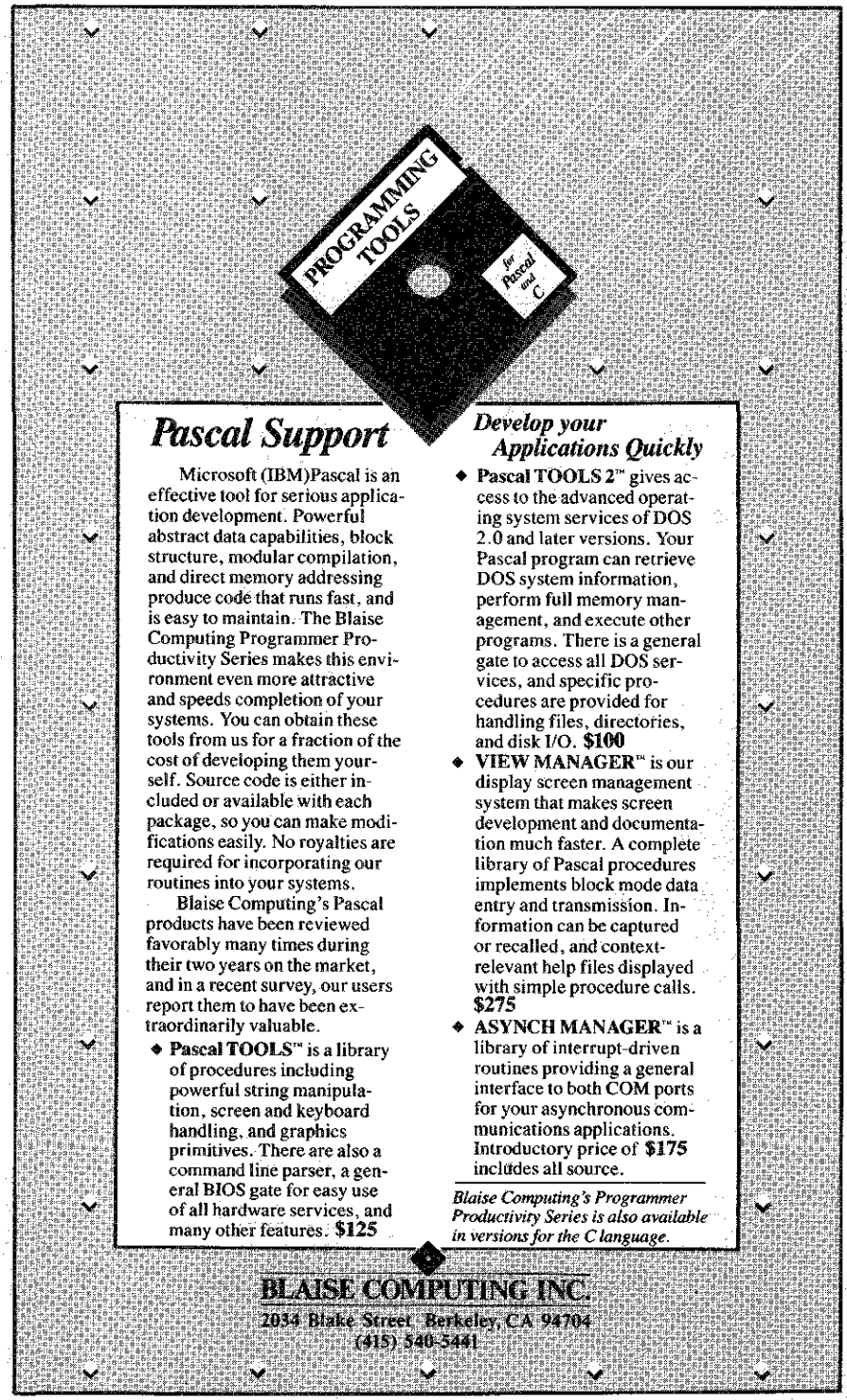
Like those 3.5 inch disk drives in the MAC for your IBM PC? Well, if you have standard 5.25 inch drives (who doesn't?) in your PC, you can plug in Mitsubishi 3.5 inch drives DIRECTLY! You can obtain these drives from a company called Manzana (805) 968-1387. The 360K drives are selling at $215; 720K drives are also available.

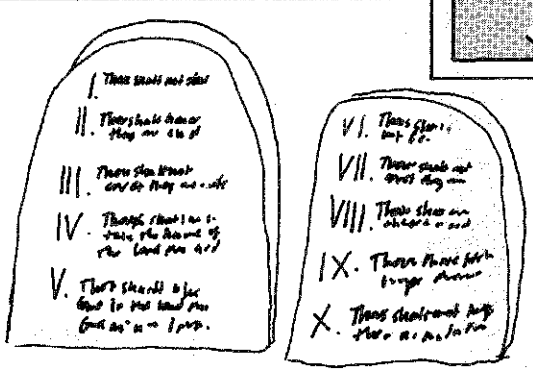
Geesham Corp., (314) 449-2284, has available a carrying case with integral 9 inch monochrome monitor that converts your IBM PCjr into the PCjr Portable.

Micron Technology in Boise, Idaho, is selling 64K DRAMS at $1.95/chip at 150 nanoseconds and $1.85/chip at 200 ns. This is the lowest in the industry. Typical prices from other vendors are around $3.00 to $4.00 per chip.

Want a fast printer? Output Technology in Spokane, Washington, has unveiled a 700 cps (!) matrix printer that will list at $1495. Correspondence-quality print is done at a mere 350 cps. In addition, these printers have Epson emulation in firmware.

Memorable quote of the month: At the Northeast Computer Fair in Boston, Dr. Alan C. Kay said, "BASIC . . . is the Laverne and Shirley of computer science." This was in regard to BASIC surviving nearly two decades of use. Dr. Kay, in speaking about state-of-the-art human interfaces with a computer, quoted sci-fi author Robert Heinlein and said, "The best weapon we have is the one between our ears . . . so long as it's loaded." □


PROGRAMMING TOOLS
for Pascal and C
Pascal Support
Microsoft (IBM)Pascal is an effective tool for serious application development. Powerful abstract data capabilities, block structure, modular compilation, and direct memory addressing produce code that runs fast, and is easy to maintain. The Blaise Computing Programmer Productivity Series makes this environment even more attractive and speeds completion of your systems. You can obtain these tools from us for a fraction of the cost of developing them yourself. Source code is either included or available with each package, so you can make modifications easily. No royalties are required for incorporating our routines into your systems.
Blaise Computing's Pascal products have been reviewed favorably many times during their two years on the market, and in a recent survey, our users report them to have been extraordinarily valuable.
◆ Pascal TOOLS™ is a library of procedures including powerful string manipulation, screen and keyboard handling, and graphics primitives. There are also a command line parser, a general BIOS gate for easy use of all hardware services, and many other features. $125
Develop your Applications Quickly
◆ Pascal TOOLS 2™ gives access to the advanced operating system services of DOS 2.0 and later versions. Your Pascal program can retrieve DOS system information, perform full memory management, and execute other programs. There is a general gate to access all DOS services, and specific procedures are provided for handling files, directories, and disk I/O. $100
◆ VIEW MANAGER™ is our display screen management system that makes screen development and documentation much faster. A complete library of Pascal procedures implements block mode data entry and transmission. Information can be captured or recalled, and context-relevant help files displayed with simple procedure calls. $275
◆ ASYNCH MANAGER™ is a library of interrupt-driven routines providing a general interface to both COM ports for your asynchronous communications applications. Introductory price of $175 includes all source.
Blaise Computing's Programmer Productivity Series is also available in versions for the C language.
BLAISE COMPUTING INC.
2034 Blake Street Berkeley, CA 94704
(415) 540-5441



The First Documentation

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DOCTOR C'S TIPS - POINTERS TO FUNCTIONS

by Rex Jaeschke

Rex Jaeschke is a consultant and editor of The C Journal. He regularly writes articles and columns on the C language and other computer-related topics for U.S. and European publications. *h Copyright by Rex Jaeschke, 1984. All rights reserved.*

Most books on C either ignore pointers to functions completely or they make passing comment by regurgitating the simple examples shown in Kernighan and Ritchie (K&R). Although having a pointer to a function may seem a 'nice but useless" capability, it can in fact save considerable coding in certain circumstances. Before we look at a useful application, let's take a moment to show exactly how pointers to functions can be defined and used.

```
/* ptrfun1.c - introduction to ptrs to functions
(C) Rex Jaeschke, 1984. All rights reserved */

main()
{
      char *mask,*str1,*str2;

      mask = "string 1 = %s string 2 = %s\n";
      str1 = "Monday";
      str2 = "December";
      printf (mask,str1,str2);
      sub (printf,mask,str1,str2); /* pass ptr to printf */
}

sub (funct,mask,str1,str2)
int (*funct)(); /* funct is a ptr to a function
                   which returns an int */
char *mask,*str1,*str2;
{
       (*funct) (mask,str1,str2);
}
```

which produces the following output:

```
string 1 = Monday string 2 = December
string 1 = Monday string 2 = December
```

The call to function sub in main uses the function name printf as its first argument. Now, as printf has already been used as a function in the previous statement, the compiler knows to generate the correct code for this argument, that is, to pass the address of the function printf. Try removing the call to function printf and compile and link. You should get an error and one which we'll discuss in the next example.

In function sub, the declaration int (*funct)(); says that the argument funct is a pointer to a function which returns an integer. Note the difference between this declaration and int *funct() which says that funct is a function that returns a pointer to an integer.

As funct is a pointer to a function, (*funct) means "use the contents of the pointer funct" just as it would for any other type of pointer. In this case, this means, "call the function pointed to by funct." Therefore, the statement (*funct) (mask,str1,str2); has the same effect as printf(mask,str1,str2); and this is demonstrated by the fact that both statements generate the same output.

OK, enough regurgitation. Let's look at a more realistic example. Suppose a program is to handle a number of different transaction or record types in the range 1 - 20. Let us assume that we will call a record type-specific function to process each transaction as it arrives. The traditional ways to handle this are by using a monstrous if/else or case construct.

Well, assembler programmers might say, "Why not have a table of function names that can be indexed by transaction type? This way, the driving program can be oblivious to the type of transaction received and the corresponding processing function will be invoked much more quickly than if a case or if/else construct were used."

The following example assumes three possible transaction types, namely 0, 1, and 2. whose corresponding processing functions are sub0, sub1 and sub2.

```
    /* ptrfun2.c - storing function ptrs in an array
(C) Rex Jaeschke, 1984. All rights reserved */

main()
{
        int sub0(),sub1(),sub2();/* 3 dummy functions */
        int (*funct)();          /* single function ptr */
        int (*funptr[3])();      /* array of function ptrs */
        int i;                   /* index variable */

        funct = sub0;            /* point to function sub0 */
        (*funct) ();             /* call sub0 */

        funptr[0] = sub0;        /* setup ptr array */
        funptr[1] = sub1;
        funptr[2] = sub2;

        for (i = 0; i < 3; ++i) /* call sub0, sub1, sub3 */
                (*funptr[i]) ();
}

sub0()
{
        printf ("Inside function sub0\n");
}

sub1()
{
        printf ("Inside function sub1\n");
}

sub2()
{
        printf ("Inside function sub2\n");
}
```

Example ptrfun2 produces the following output:

```
Inside function sub0
Inside function sub0
Inside function sub1
Inside function sub2
```

As sub0 as been declared to be a pointer to a function, the statement funct = sub0; assigns the address of function sub0

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

to funct and once again sub0 is invoked indirectly using (*funct)();. Note that if sub0 was not declared at all, funct = sub0; would cause sub0 to be incorrectly interpreted as a global int variable (or possibly be flagged by the compiler as an error). In this case, such a global variable would not be defined anywhere else in the program so an "unsatisfied external" error would result at link-time. As functions sub0, sub1 and sub2 must be explicitly declared in main they must be given a pointer object type, in this case, int. Even though these functions do not return a value they must have a type to satisfy the language declaration syntax requirements. Of course, if such a function returned a value of type other than int, the appropriate type should be used instead.

The declaration int (*funptr[3])(); says that funptr is an array of 3 pointers to functions which return integers while the statement funptr[0] = sub0; assigns the address of function sub0 to the first array element. As funptr is an array, its elements can be referenced by subscript such as in (*funptr[i])(); which invokes the function pointed to by element i. And hey presto! We have an array of function names which can be called by knowing the array index (which could be a transaction type).

```
/* ptrfun3.c - initializing an array of function ptrs
(C) Rex Jaeschke, 1984. All rights reserved */

main()
{
      int sub0(),sub1(),sub2(); /* 3 dummy functions */
      static int (*funptr[3])() /* array of function ptrs */
            = {sub0,sub1,sub2};
      int i;                    /* index variable */

      for (i = 0; i < 3; ++i) /* call sub0, sub1, sub3 */
            (*funptr[i]) ();
}
```

which produces the following output:

```
Inside function sub0
Inside function sub1
Inside function sub2
```

Example ptrfun2 defined the function pointer table at run-time, a rather expensive approach considering that the function names and their order in the table will most likely remain constant throughout the program. This example shows how to define the table as a static, local array using static int (*funptr[3])() = (sub0,sub1,sub2);. This definition and initialization is similar to that for any other array. Note that the function names must not be in quotes and that they must be explicitly declared or invoked prior to this statement so that they are correctly recognized as functions.

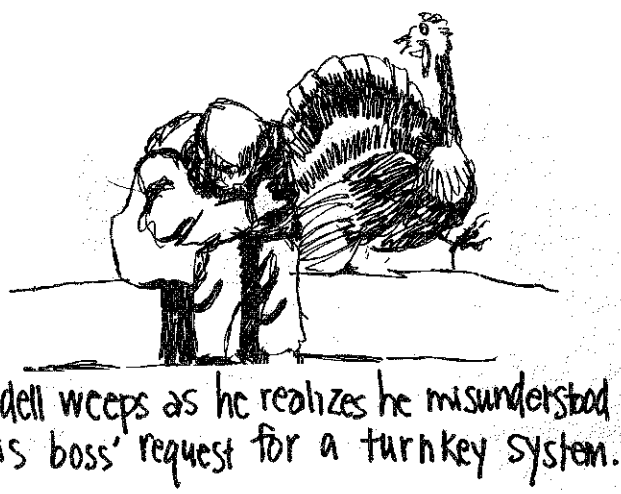
```
/* ptrfun4.c - more on initializing an array of function ptrs
(C) Rex Jaeschke, 1984. All rights reserved */
int sub0(),sub1(),sub2();
int (*funptr[3])() = {sub0,sub1,sub2};
main()
{
        int i;                    /* index variable */

        for (i = 0; i < 3; ++i) /* call sub0, sub1, sub3 */
              (*funptr[i]) ();
}
```

This example is quite similar to ptrfun3 except that the pointer name table is global. Note again that the functions referenced in the initialization must have been declared previously.

One last thing. Just what exactly is a pointer to a function and when is it known? Perhaps that's best explained by looking at the 8088/86 code generated by the relevant parts of ptrfun1.c above. The function pointer code generated by the compiler is based on the offset of the function from some particular base address, neither of which is yet known. The actual addresses of these are resolved at link-time when the function is finally assigned an actual starting address relative to the executable module's base or some other segment.

```
...
printf (mask,str1,str2);
sub (printf,mask,str1,str2);     /* pass ptr to printf */

        PUBLIC printf_
        PUBLIC sub_

main_:  PUSH BP
        ...
        MOV AX,OFFSET printf_
        PUSH AX
        CALL sub_
        ...
        POP BP
        RET

sub (funct,mask,str1,str2)
int (*funct)();
char *mask,*str1,*str2;
{
        (*funct) (mask,str1,str2);


sub_:   PUSH BP
        MOV BP,SP
        PUSH WORD [BP+10]
        PUSH WORD [BP+8]
        PUSH WORD [BP+6]
        CALL WORD [BP+4]          ;jump to function pointed to
                                  ;by 1st argument
        MOV SP,BP
        POP BP
        RET
        END
```

There now, that's not so complicated after all. You may be surprised at how much simpler (and faster) your transaction processing code can be.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE ASK/BATCH FILE UTILITY

by Dan Rollins

I'm no batch file fanatic, but I have a written a few batch files to simplify my programming environment. And as a consultant, I have found them invaluable for setting up "turn-key" systems for clients. For instance, a batch file named WP.BAT can initialize a serial printer, set the default sub-directory, and invoke the word processor program. New secretaries prefer two-letter commands. In fact, the most foolproof technique is to format a system diskette with a customized AUTOEXEC batch (even when using a hard disk). Then my instructions are as easy as "insert this disk and flip the red switch."

I have discovered several simple batch file techniques that are not particularly obvious. The first trick is to redirect the output of the ECHO command. The command

```
ECHO.>LPT1 ^O
```

where ^O is as ASCII 15, will set an EPSON printer into compressed mode. Place this in a file named COMPRESS.BAT and your clients will love you. The ECHO command can also be used to set screen attributes when you have installed the ANSI.SYS console driver.

One limitation of batch files is that they can't be nested. If one batch invokes a second batch, control won't return to the first. You can get around this by using the COMMAND command. For instance:

```
rem BATCH_A.BAT
command /c batch_b
echo back in BATCH_A

rem BATCH_B.BAT
echo executing BATCH_B
```

This technique could be used when you want to run a program that is "filtered" by a batch file. For instance, I always rename the FORMAT.COM file as DESTROY.COM, then have a batch file named FORMAT.BAT that will never format the hard disk. It's also a way to extend the one-command limit of the IF conditional batch operations.

It could be argued that batch files should be just that — a file composed of a noninteractive sequence of DOS commands. But it's not hard to find batch file applications that would benefit from a little user interaction.

The conditional operations that are available are extremely limited. About the only useful one is the IF EXIST command, which can be used as part of a sequence that copies a program file to RAM disk before invoking it. Incidentally, DOS 3.x allows path names in the IF EXIST command, making it slightly more useful. Alas, it still can't be used to find out if a system has two physical floppy disk drives.

About the only use for the

```
IF string1 (GARY -- DOUBLE EQUAL) string2 command
```

is to abort a batch file when a user forgets to to enter a required parameter. For example,

```
IF .%1. (GARY -- DOUBLE EQUAL) .. goto abort
  .
  .
  .
:abort
echo Missing parameter
```

Also, this command would be easier to use if it weren't case-sensitive on the string comparison.

IBM's inclusion of IF ERRORLEVEL among the batch commands is the kind of thing that intrigues me. Most of the PC-DOS commands don't even set the ERRORLEVEL, so this feature is all but ignored in the DOS manual. In fact, "error level" is a misnomer because no "error" has to occur — it's based on the exit code that may be passed back to DOS by a program that returns via DOS service 4Ch. But I suppose EXITCODELEVEL is a little verbose.

Anyway, assembler and C programs can easily set the exit code. This makes it possible to get numeric ERRORLEVEL input into an executing batch file.

Assembler listing 1, ASK.COM, is a program that uses this trick. It's operation is quite simple. It displays the prompt defined on its parameter line, then waits for a keystroke. If that keystroke is found in its valid-input list, then it returns to DOS with a corresponding exit code.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

As you can see from the comments, some of the keystrokes share the same exit codes. For instance, < Y > and < y > and < 1 > and < F1 > all return an exit code of 1. My idea was to be able to accept a number key from any place on the keyboard — top row, left-side function keys, or numeric keypad. You may change the lookup tables if you want to different values for Y, N, and Q. And you can add new keystrokes and exit codes easily.

The comments include a simple batch file that tests for a Yes/No answer. I am also including a more complex batch file that displays a short menu — the type of thing that a nonexpert might like to see on powering up the old XT.

The ^I in some of the ECHO commands is just a TAB character (ASCII 9). This is a simple way to ECHO a blank line to the screen. The G is the BEL character, which is an effective attention-getter.

If you look closely, you'll find another trick of the trade. I try to avoid displaying confusing messages on the screen, so when I invoke the MODE command, I route its output to the NUL device. Simple and effective.

One final note: batch files are notoriously slow. But if you copy a batch file to RAM-disk before executing it, the speed is acceptable.

I hope that some readers can make use of the these tidbits of wisdom. I have found the ASK program to be a real boon in certain applications. Of course when TopView takes hold, I suspect that complex batch files will go the way of tiny BASIC.

C + C UTILITY LIBRARY = PRODUCT

• We have over 200 *complete, tested, and, documented functions.* All source code and demo programs are included.

• The library was specifically designed for software development on the IBM PC, XT, AT and compatibles. There are *no royalties.*

• Over 95% of the source code is written in C. *Experienced programmers* can easily "customize" functions. *Novices* can learn from the thorough comments.

We already have the functions you are about to write

Concentrate on software development—not writing functions.

THE C UTILITY LIBRARY includes:

• Best Screen Handling Available • Windows • Full Set of Color Graphics Functions • Better String Handling Than Basic • DOS Directory and File Management • Execute Programs, DOS Commands and Batch Files • Complete Keyboard Control • Extensive Time/Date Processing • Polled ASYNC Communications • General DOS/BIOS gate • And More •

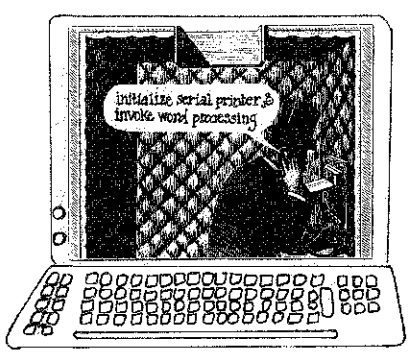
• The Library is compatible with: Lattice, Microsoft, Computer Innovations, Mark Williams and DeSmet.

C Compilers: Lattice C—$349, Computer Innovations C86—$329; Mark Williams C—$449.

C UTILITY LIBRARY $149

Order direct or through your dealer. *Specify compiler* when ordering. Add $4.00 shipping for UPS ground, $7.00 for UPS 2-day service. NJ residents add 6% sales tax. Master Card, Visa, check or P.O.

ESI ESSENTIAL SOFTWARE, INC
P.O. Box 1003 Maplewood, New Jersey 07040 914/762-6605



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
;-------- ASK.ASM -------------
;
; Batch file utility sets ERRORLEVEL according to user input
;
; syntax:
;
; ASK prompt-string
;
; The prompt-string is displayed followed by 1 space, then the program
; pauses till a key is pressed.  If the key is acceptable, it
; is displayed (followed by CR) and ERRORLEVEL is set.  Otherwise, a beep is
; sounded, and the program continues to wait.
;
;    valid     | returned
;   replies    | ERRORLEVEL
;--------------+---------------------
;  Esc q Q     |    0
;  1 Y y F1    |    1
;  2 N n F2    |    2
;  3     F3    |    3
;  .     .     |    .
;  .     .     |    .
;  9     F9    |    9
;
; example usage in a batch file:
;
;    echo off
;    :start   BASCOM compile & link with optional re-edit
;    bascom %1;
;    ask Re-edit source code (Y/N)?
;    if errorlevel 2 goto continue
;    rem  control falls through when user answers Y
;    edlin %1.BAS
;    goto start
;    :continue   this branch taken when user answers N
;    link %1;


com_seg  segment para
         assume cs:com_seg, ds:com_seg, es:com_seg

         org     100H
ask      proc    near
         jmp     start

; These tables have three bytes per element:
; 1) the ASCII input value
; 2) the exit code returned to DOS
; 3) the character to display

normal_keys label byte   ;non-extended ASCII table

         db 'Q',0,'Q', 'q',0,'Q',  27,0,'Q'    ;last is Esc
         db 'Y',1,'Y', 'y',1,'Y', '1',1,'1'
         db 'N',2,'N', 'n',2,'N', '2',2,'2'
         db '3',3,'3', '4',4,'4', '5',5,'5'
         db '6',6,'6', '7',7,'7', '8',8,'8', '9',9,'9'

NK_LEN   equ ($ - normal_keys)/3          ;count of 3-byte elements in table

fn_keys label byte   ;extended ASCII

         db 59,1,'1', 60,2,'2', 61,3,'3', 62,4,'4', 63,5,'5' ;fn keys
         db 64,6,'6', 65,7,'7', 66,8,'8', 67,9,'9'
         db 79,1,'1', 80,2,'2', 81,3,'3', 75,4,'4', 76,5,'5' ;num pad
         db 77,6,'6', 71,7,'7', 72,8,'8', 73,9,'9'

FK_LEN   equ ($ - fn_keys)/3          ;count of 3-byte elements in table

bad_ver_msg  db 'Wrong DOS version',0dh,0ah,'$'

start:
         mov     ah,30h        ;check if DOS 2.x
         int     21h
         cmp     al,2                    ; at least version 2.0?
         jae     a_10                    ;  yes, go
         mov     dx,offset bad_ver_msg   ;  no,
         mov     ah,9                    ;   display message
         int     21h                     ;     and
         int     20h                     ;      exit
a_10:
         mov     si,80h                  ; point to parameter string
         mov     cl,[si]                 ; get length
         inc     si
         mov     ch,0
         add     si,cx                   ; find end of prompt string
         mov     byte ptr [si],' '       ; add a space
         mov     byte ptr [si+1],'$'     ; make compatible with DOS service 9

         mov     dx,82h                  ;point to first character
         mov     ah,9
         int     21h                     ;display the prompt-string

again:                           ;--- re-entry point after unknown input

         mov     di,offset normal_keys ;assume non-extended (normal) key
         mov     cx,NK_LEN

         mov     ah,7
         int     21h               ;wait for a key
         cmp     al,0              ;extended ASCII?
         jne     a_20              ;  no, skip
         mov     ah,7              ;  yes,
         int     21h               ;  fetch the scan code into AL
         mov     di,offset fn_keys ;  and set up for extended ASCII table
         mov     cx,FK_LEN
a_20:
         scasb                      ;is AL in key table?
         jz      a_30               ; yes, we're done
         inc     di                 ; no, get past return code in table
         inc     di                 ;     and get past display byte
        loop    a_20                ;     and try next element

         call    beep               ; not found , so beep
         jmp     again              ;  and restart
a_30:
         mov     dl,[di+1]          ; it was found
         mov     ah,2               ; so display the character
         int     21H
```

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        call    crlf                ;and linefeed carriage-return

        mov     al,[di]             ;fetch the return code

        mov     ah,4ch
        int     21H                 ;return AL as error level

ask     endp


;==========================
; BEEP near procedure
; sounds a beep by 'printing' an ASCII 7 (BEL character)
; preserves all registers

beep    proc    near
        push    ax
        push    dx
        mov     ah,2
        mov     dl,7        ;BEL character
        int     21h
        pop     dx
        pop     ax
        ret
beep    endp


;==========================
; CRLF near procedure
; displays a linefeed and carriage-return
; preserves all registers

crlf    proc    near
        push    ax
        push    dx
        mov     dl,0dh
        mov     ah,2
        int     21h

        mov     dl,0ah
        mov     ah,2
        int     21h

        pop     dx
        pop     ax
        ret
crlf    endp

com_seg ends
        end     ask
```

--------- MENU.BAT ---

```
echo off
rem *** powerup menu for AAAA Managerial Consultants, Inc.***

:START
cls
echo ============= M E N U =============
echo =                                    =
echo =      [1]  Lotus 1-2-3              =
echo =                                    =
echo =      [2]  Wordstar                 =
echo =                                    =
echo =      [3]  Manage-Man Menu          =
echo =                                    =
echo =                                    =
echo =      [Q]  Quit and exit to DOS     =
echo =                                    =
echo ====================================
echo ^I
echo ^I
ask Select option or Quit (1,2,3, or Q):
if not errorlevel 1 goto QUIT
if not errorlevel 2 goto 123
if not errorlevel 3 goto WS
if not errorlevel 4 goto MM
echo ^I
echo ^G invalid response! ^G
goto START


:MM
cd \mm
manager
cd \
goto START


:WS
cd \wp\ws
mode com1:12,n,8,1,p >NUL
mode lpt1:=com1      >NUL
ws
cd \
goto START

:123
cd \lotus
123
cd \
goto START

:QUIT
cls
echo ======= Exiting to PC DOS ========
echo == enter MENU to return to menu ==
echo ================================== □
```

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

INDEX FILES

by Ronald Moak

In Part I of this article we looked at the structure of a simple index file. Index files of this type are often considered poor man's ISAM structures. While not as elegant as true ISAMs, they are nevertheless useful in numerous programming cases. In this article we will look at the code needed to move data into and out of the main data file, using the index file as a reference to the main file.

In Part I we ended the article with the subroutine needed to open the database files. I have included the open routine here with an additional list of constants to be placed at the beginning of the program. In practice once these subroutines are completely debugged, they should be placed in a secondary file and referenced with an include statement in the main program. This makes the routines easily available to any program written to access the database.

```
'*********************************************************
'*
'*   Constants
'*
'*********************************************************
NDX.FILE% = 2
DAT.FILE% = 1
HEADER.REC% = 1
OVER.MAX% = 20
TRUE% = 1
FALSE% = 0

'*********************************************************
'*
'*   Open database and index files
'*
'*********************************************************
10000
open "INVENTOR.DAT" AS DAT.FILE% len = 79
open "INVENTOR.NDX" AS NDX.FILE% len = 10

field DAT.FILE%, 2 as FDAT.RECNUM$      'header record data file

field DAT.FILE%, 1  as  FDELETE$                    'data file record
                 8  as  FPART.NUMBER$                _
                 40 as  FPART.DESCRIPTION$           _
                 6  as  FSUPPLIER.CODE$              _
                 2  as  FQUANTITY.ON.HAND$           _
                 2  as  FREORDER.QUANTITY$           _
                 8  as  FPURCHASE.PRICE$             _
                 8  as  FSELLING.PRICE$              _
                 2  as  FVOLUME.DISCOUNT.PERCENT$    _
                 2  as  FVOLUME.DISCOUNT.MIM.QT$


field NDX.FILE%, 2 as FNDX.RECNUM$,  _ 'header record index file
                 2 as FNDX.SORTNUM$

field NDX.FILE%, 8 as FXPART.NUMBER$ _ 'data record index file
                 2 as FXREC.NUMBER$

field NDX.FILE%, 10 as TEMP.BUFFER$

get DAT.FILE%,1
DAT.RECNUM% = cvi(FDAT.RECNUM$)

get NDX.FILE%,1
NDX.RECNUM%  = cvi(FNDX.RECNUM$)
NDX.SORTNUM% = cvi(FNDX.SORTNUM$)
return
```

The first function of the open database subroutine is the opening of the index and data files. The second function is to initialize several variables used by the rest of the routines in accessing the database. The following variables are initialized in the open database routine:

DAT.RECNUM% — contains the total number of records within the database less one record that contains the header information.

NDX.RECNUM% — contains the total number of records within the index file less one record that contains the header information. In practice this variable will be equal to DAT.RECNUM%.

NDX.SORTNUM% — contains the number of sorted records contained within the index file. The number of overflow records equals NDX.RECNUM% – NDX.SORTNUM%.

Now that we have opened the database, let's add some information to it. When adding new records, we place them at the end of the file. This subroutine is called when we wish to add a new record. We will have another subroutine to write out data to an existing record. Because no checking is performed by the subroutines to determine the correct order called, it is up to the programmer to insure correct program flow. For example, before calling the append record routine you should first call the find record subroutine (illustrated later). If no record exists, the record may be appended to the file. By searching before appending we can better guarantee that there will be no duplicate records with the same key.

In order to add a record, we need to perform several operations. This routine must update the header in the index file, add a record to the index file that points to the data file, update the header in the data file, and write the new record to the data file. The final function of the append routine is to test for the number of overflow records contained within the index. If the number of records is greater than the specified number, the index file is automatically sorted.

If all of this seems a little confusing, hold on — it will become clearer (at least I hope so). To update the two header records we simply add one to the two variables that contain the number of records in the database (DAT.RECNUM and NDX.RECNUM) and write the records to disk. In our routines we first read the contents of the header record before updating it. This step insures the integrity of the values, unfortunately, it also increases the time needed to append the record. If the database required a lot of rapid appending, this step could be eliminated because we set the values of the variables when we opened the file. However, for most data entry this does not sufficiently degrade performance.

Now, to add the index record, create one by placing the value of the variable DAT.RECNUM U 1 into a pointer variable in the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

index record, FXREC.NUMBER$. This may be done with the following statement.

```
lset FXREC.NUMBER$ = mki$(DAT.RECNUM% + 1)
```

The part number is first uppercased then moved into the key variable and the record is written. All that is left now is to place the rest of the data values into the file buffer and write the record to disk. The append record subroutine performs this operation by setting a global record pointer and calling another subroutine that performs the actual write record operation.

```
'*********************************************************
'
'   Append database
'
'*********************************************************
000
t DAT.FILE%, HEADER.REC%                    ' get header information
t NDX.FILE%, HEADER.REC%                    '
T.RECNUM% = cvi(FDAT.RECNUM$)               '
X.RECNUM% = cvi(FDAT.RECNUM$)               '
et FDAT.RECNUM$ = mki$(DAT.RECNUM% + 1)     ' update header
et FNDX.RECNUM$ = mki$(DAT.RECNUM% + 1)     '
t DAT.FILE%, HEADER.REC%                    ' write headers
t NDX.FILE%, HEADER.REC%                    '
MP.PART$ = PART.NUMBER$                     ' uppercase the part number
r FR.I% = 1 to len(TEMP.PART$)              ' store only uppercase
 CH$ = mid$(TEMP.PART$,FR.I%,1)             ' characters in index
 if CH$ > 96 and CH$ < 123 then             '
    CH$ = str$(asc(CH$)-32)                 '
    mid$(TEMP.PART$,FR.I%,1) = CH$          '
xt FR.I%                                    '
et FXPART.NUMBER$ = TEMP.PART$              ' initialize index
et FXREC.NUMBER$  = FDAT.RECNUM$            ' record
OB.REC.NUMBER% = cvi(FDAT.RECNUM$)          ' init global variable
t NDX.FILE%, GLOB.REC.NUMBER% + HEADER.REC% ' write index record
sub 12000                                   ' write data record
X.SORTNUM% = cvi(FNDX.SORTNUM$)
 NDX.RECNUM% - NDX.SORTNUM% > OVER.MAX% then ' test overflow max and
 gosub 13000                                ' gosub sort index file
turn
```

The reason for isolating the writing of the data to a separate routine is to allow the one write record routine to do double duty. The write record routine may be called from the append database routine or called directly to update the last record read.

The write routine is a bare-bones routine and provides no error checking. At least two checks should be applied to a production version. The first check insures that the global record number is no greater than the number of records in the file plus one. The second check is to insure the record number is greater than zero. A record number of zero will trash the header record.

```
'*********************************************************
'
'   Write database record
'
'*********************************************************
12000
lset FPART.NUMBER$              = PART.NUMBER$
lset FPART.DESCRIPTION$         = PART.DESCRIPTION$
lset FSUPPLIER.CODE$            = SUPPLIER.CODE$
lset FQUANTITY.ON.HAND$         = mki$(QUANTITY.ON.HAND)
lset FREORDER.QUANTITY$         = mki$(REORDER.QUANTITY)
lset FPURCHASE.PRICE$           = mks$(PURCHASE.PRICE)
lset FSELLING.PRICE$            = mks$(SELLING.PRICE)
lset FVOLUME.DISCOUNT.PERCENT$ = mki$(VOLUME.DISCOUNT.PERCENT)
lset FVOLUME.DISCOUNT.MIN.QT$  = mki$(VOLUME.DISCOUNT.MIN.QT)
put DAT.FILE%, GLOB.REC.NUMBER% + HEADER.REC%
return
```

Once the record is written to the data file, a test is performed to determine if the index needs to be sorted. The size of the overflow area depends on a number of factors. The fewer records in the overflow, the faster the search, because fewer records need to be sequentially searched to find the desired record. The flip side to that is more time spent sorting. If more time is spent in data retrieval than in adding records, a smaller overflow area is desired. The size of the overflow depends on your application.

The sort routine listed below is a very slow bubble sort that would make dBase II's sort seem fast. It is included here only to illustrate the concepts. A good sort algorithm may be obtained from a number of computer books.

```
'*********************************************************
'*
'*   Sort index file
'*
'*********************************************************
13000
locate 25,1
print "Sorting Index File !!!";
for SIF.I% = 2 to NDX.RECNUM% -1
   get NDX.FILE%,SIF.I%
   TEMP.BUF1$ = TEMP.BUFFER$
   for SIF.J% = SIF.I% + 1 to NDX.RECNUM%
       get NDX.FILE%,SIF.I%
       TEMP.BUF2$ = TEMP.BUFFER$
       if right(TEMP.BUF1$,8) > right(TEMP.BUF2$,8) then
          swap TEMP.BUF1$, TEMP.BUF2$
          lset TEMP.BUFFER$ = TEMP.BUF1$
          put NDX.FILE%,SIF.I%
          lset TEMP.BUFFER$ = TEMP.BUF2$
          put NDX.FILE%,SIF.J%
   next SIF.J%
next SIF.I%
locate 25,1
print "                         ";

lset FNDX.SORTNUM$ = mki$(NDX.RECNUM%)    ' Reset the sort number
put NDX.FILE%,HEADER.REC%                 '
return
```

Now that we have the means to put information into the database and maintain it in a sorted order, we need a routine to locate specific information from the database. We have designated PART.NUMBER$ as the key piece of information in which to locate a record. In our find record routine we need to initialize PART.NUMBER$ prior to calling the subroutine.

The initial function of the subroutine is to perform a binary search of the sorted records. If no matching record is found the routine will sequentially search any records contained in the overflow. If a record is not found, the routine will set an error flag (FILE.ERROR); otherwise the global record variable will be set to the record number of the data record in the data file. To actually read the record into memory, the read record subroutine is called.

```
'*********************************************************
'*
'*   Find record
'*
'*********************************************************
14000
FILE.ERROR% = FALSE%
FR.GOTIT% = FALSE%
get NDX.FILE%, HEADER.REC%                  '
NDX.RECNUM%  = cvi(FNDX.RECNUM$)            '
NDX.SORTNUM% = cvi(FNDX.SORTNUM$)           '

if NDX.RECNUM% = 0 then  14099
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MasterFORTH

Portable programming environment

Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest. If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.

MasterFORTH is a state-of-the-art implementation of the Forth computer language. Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in macro assembler to make it even faster. MasterFORTH's relocatable utilities, transient definitions, and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint, and trace your way through most programming problems. A string package, file interface, and full screen editor are all standard features.

MasterFORTH exactly matches the Forth-83 Standard dialect described in *Mastering Forth* by Anderson and Tracy (Brady, 1984). The standard package includes the book and over 100 pages of supplementary documentation.

MasterFORTH standard package	
Macintosh	$125
IBM PC and PC Jr. (MS DOS 2.1)	125
Apple II, II+, IIe, IIc (DOS 3.3)	100
CP/M 2.X (in several formats)	100
Commodore 64	100
Extensions	
Floating Point (1984 FVG standard)	$40
Graphics (Apple II series)	40
Module relocator (with utility sources)	60
Printed source listing (each)	35
Publications	
Mastering Forth (additional copies)	$18
Thinking Forth by Leo Brodie	16
Forth-83 International Standard	15
Rochester Bibliography, 2nd ed.	15
1984 Rochester Conference	25
1984 J1 of Forth Appl. & Res. 2(2)	15
1983 FORML Conference	25

MICROMOTION

12077 Wilshire Blvd., #506
Los Angeles, CA 90025
(213) 821-4340


```
KEY.PART$ = PART.NUMBER$
for FR.I% = 1 to len(KEY,PART$)
   CH$ = mid$(KEY.PART$,FR.I%,1)
   if CH$ > 96 and CH$ < 123 then
      CH$ = str$(asc(CH$)-32)          :_
      mid$(KEY.PART$,FR.I%,1) = CH$
next FR.I%

if NDX.SORTNUM% = 0 then 14010          'skip if no sorted recs.

'---------------- SEARCH BINARY ----------------------
FR.MAX% = NDX.SORTNUM%
FR.MIN% = 1

while (FR.MIN% < FR.MAX%) or (FR.GOTIT% = FALSE%)

    FR.MID% = (FR.MAX% - FR.MIN%) / 2
    get NDX.FILE%,FR.MID% + HEADER.REC%
    REC.PART$ = FXPART.NUMBER$

    if REC.PART$ = KEY.PART$ then                          _
        FR.GOTIT% = TRUE%                                  :_
        GLOB.REC.NUMBER% = cvi(FXREC.NUMBER$)              _
    else if REC.PART$ < FXPART.NUMBER$                     _
        FR.MAX% = FR.MID%                                  _
    else                                                   _
        FR.MIN% = FR.MID%

wend

14010 '------------ SEARCH SEQUENTIAL ---------------
if (FR.GOTIT% = TRUE%) or (NDX.SORTNUM% = NDX.RECNUM%) then 14099

for FR.I% = NDX.SORTNUM% to NDX.RECNUM%

    get NDX.FILE%,FR.I% + HEADER.REC%
    REC.PART$ = FXPART.NUMBER$

    if REC.PART$ = KEY.PART$ then                          _
        FR.GOTIT% = TRUE%                                  :_
        GLOB.REC.NUMBER% = cvi(FXREC.NUMBER$)              :_
        FR.I% = NDX.RECNUM%

next FR.I%
14099''--------------EXIT ROUTINE ----------------------
if FR.GOTIT% = FALSE% then  _
   FILE.ERROR% = TRUE%

return
```

Now that we know that the record exists and the record pointer has been set we need one more subroutine to read the information from the database. The read record routine is similar to the write record subroutine in that it expects the record pointer to be set prior to being called. As with the write routine this routine, needs a few checks to make sure the records being read are within the logical limits of the file.

```
'*********************************************************
'*
'*   Read database record
'*
'*********************************************************

15000
get DAT.FILE%, GLOB.REC.NUMBER% + HEADER.REC%
PART.NUMBER$              =  FPART.NUMBER$
PART.DESCRIPTION$         =  FPART.DESCRIPTION$
SUPPLIER.CODE$            =  FSUPPLIER.CODE$
QUANTITY.ON.HAND          =  cvi$(FQUANTITY.ON.HAND$)
REORDER.QUANTITY          =  cvi$(FREORDER.QUANTITY$)
PURCHASE.PRICE            =  cvs$(FPURCHASE.PRICE$)
SELLING.PRICE             =  cvs$(FSELLING.PRICE$)
VOLUME.DISCOUNT.PERCENT   =  cvi$(FVOLUME.DISCOUNT.PERCENT$)
VOLUME.DISCOUNT.MIM.QT    =  cvi$(FVOLUME.DISCOUNT.MIM.QT$)
return
```

The final two subroutines we will cover are utility routines. A delete routine is added to allow you to flag certain records that are no longer needed. These records are only flagged at this point. If you wanted you could easily modify the routine to un-delete the record.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The second utility routine packs and reindexes the database and removes the deleted records. This routine is useful in cases where the file that contains the index gets trashed and you need to rebuild the index. It is also useful when your file sizes are stretching the disk space. This routine will free up space used by deleted records.

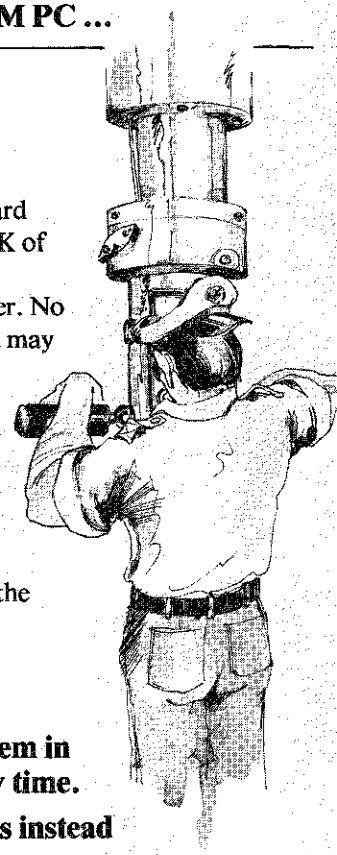
```
'*********************************************************
'*
'*   Delete database record
'*
'*********************************************************
16000
lset FDELETE$ = "D"
put DAT.FILE%, GLOB.REC.NUMBER% + HEADER.REC%
return

'*********************************************************
'*
'*   Pack and reindex database
'*
'*********************************************************
17000
RR.I% = 1                                   ' init. read rec. index
WR.I% = 1                                   ' init. write rec. index
                                            '
close DAT.FILE%, NDX.FILE%                  ' close database
KILL "INVENTOR.NDX"                         ' start with new index file
gosub 10000                                 ' open database
                                            '
while not eof(DAT.FILE%)                    '
 get DAT.FILE%, RR.I% + HEADER.REC%         '
 if FDELETE$ = "D" then 17010               ' rewrite only non-deleted
                                            ' records
    KEY.PART$ = FPART.NUMBER$               '
    for FR.I% = 1 to len(KEY.PART$)         ' uppercase index key
       CH$ = mid$(KEY.PART$,FR.I%,1)        '
                                            '
    if CH$ > 96 and CH$ < 123 then          '
       CH$ = str$(asc(CH$)-32)              '
       mid$(KEY.PART$,FR.I%,1) = CH$        '
                                            '
    next FR.I%                              '
                                            '
    lset FXPART.NUMBER$ = KEY.PART$         ' set up index record
    lset FXREC.NUMBER$  = cvi(WR.I%)        '
    put NDX.FILE%, WR.I% + HEADER.REC%      ' write index record
    put DAT.FILE%, WR.I% + HEADER.REC%      ' write data record
    WR.I% = WR.I% + 1                       ' inc. write index
                                            '
   17010                                    '
   RR.I% = RR.I% + 1                        ' inc. read index
wend                                        '
                                            '
lset FDAT.RECNUM$ = mki$(WR.I%)             ' reset header records to
lset FNDX.RECNUM$ = mki$(WR.I%)             ' new record counts
put DAT.FILE%, HEADER.REC%                  '
put NDX.FILE%, HEADER.REC%                  '
                                            '
gosub 13000                                 ' sort index file
return
```

The subroutines we have covered in this article form the basis for moving data into and out of a file controlled by an index file. In your work you will want to enhance these routines to provide additional functionality and error checking. Other routines that should probably be added include a get next or get previous record routine to process the file in a sorted format and an un-delete function to recover deleted records.

It would be fairly simple to modify these routines to be more flexible in handling different files. Use your imagination in making them more bullet proof and responsive to your needs. □

Debugging Bugging You?

Torpedo program crashes and debugging delays with debugging dynamite for the IBM PC ...



UP PERISCOPE!

First, you install the hardware.

The hardware's a special memory board that fits in a PC expansion slot. Its 16K of write-protected memory contains Periscope's resident symbolic debugger. No runaway program, however berserk it may be, can touch this memory!

Then you UP PERISCOPE.

Use Periscope's push-button break-out switch to interrupt a running program ... even when the system's hung! Periscope supports Assembly, BASIC, C and Pascal. In addition to the usual debugging capabilities, some of Periscope's features are:

Stop your system in its tracks at any time.

Use symbol names instead of addresses.

Run a program on one monitor and debug on another.

Monitor your program's execution with Periscope's comprehensive breakpoints.

Debug memory-resident programs.

Put your time to better use.

The Periscope system is $295. It carries a 30-day money-back guarantee and includes the memory board, remote break-out switch, debugger software, 100-page manual, and quick-reference card. The memory board is warranted for one year. A demonstration disk is $5.00.

System requirements for Periscope are an IBM PC, XT or Compaq, PC-DOS, 64K RAM, 1 disk drive and an 80-column monitor. For MasterCard and Visa orders only, call 800/421-5300 (ext. R96) 24 hours a day. For additional information, call 404/256-3860 from 9 AM to 5 PM Eastern Time.

Get your programs up and running;

UP PERISCOPE!

Data Base Decisions / 14 Bonnie Lane /Atlanta, GA | 30328

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SCANNING IN TURBO PASCAL

by Tom Swan

Turbo and UCSD Pascal are as close as kissing cousins. They're not the perfect couple, but neither are they strangers to each other.

When converting from one Pascal to the other, differences between the two compilers become more noticeable, particularly in the extensions added to the standard language. For example, while converting my relational data base system, PDBS, to Turbo Pascal, I was mildly disappointed by the absence of one favorite UCSD extension, the scan function.

So, you might say, just write a scan function. And so I thought, until I looked more closely at Softech Microsystems' documentation. Here is how scan is defined in UCSD Pascal:

```
FUNCTION scan( length : integer;
               partial expression,
               VAR array           ) : integer;
```

Scan is used whenever an array of bytes must be searched for a particular value, or the absence of that value. The nice thing about scan is that it can be used on any kind of array, a string, a disk buffer, or any other data type. With the scan function in place, there is no need to write specific procedures for scanning arrays having different contents.

Length, an integer passed by value, represents the number of bytes to be scanned. If length is positive, then the scan will be performed in a forward direction (toward greater indexes); if negative, then the scan goes backward (toward lesser indexes).

Partial expression is either < > ch or =ch, in exactly that form. This is not a boolean expression, but a kind of string that only the scan function knows how to read. A partial expression has no associated data type. Ch is supposed to be a character, but can be any 8-bit value n by using the chr(n) function in place of ch. For example, to scan for the value 255, you could use =chr(255) as the partial expression. To scan for all non-nulls, you would use < > chr(0).

Array, a variable parameter, may have been better named as a memory pointer, which is what it really is. Although untyped, an array is always an array of bytes, regardless of the assigned contents of the array. An arrayof 16-bit integers, in other words, would be scanned as an array of 8-bit bytes with no regard for the high and low byte ordering of the arrayed integers. Array may be indexed to start the scan at some location other than the first element of the array (the default condition).

CONVERTING SCAN TO TURBO PASCAL

Because I could think of no way to declare a partial expression data type compatible with the UCSD definition, I decided to break an expression into two parts, an opcode and a character. Here is my definition of the new Turbo scan:

```
FUNCTION scan(     limit   : integer;
                   opcode,
                   ch      : char;
              VAR source           ) : integer;
```

Limit is the same as in the UCSD version. Opcode is either '#' for "not equals" or '=' for "equals." Ch is the character, or 8-bit value, for which we are scanning.

Source is an untyped variable, allowed by Turbo Pascal, and interpreted as a simple pointer with no associated data type. Source is the equivalent of the array variable in the UCSD version. It was not possible to call this pointer "array," however, because the identifier "array" is, of course, reserved by the Pascal compiler.

Having decided upon a suitable design, as close to its UCSD relative as I could devise, the actual programming was not difficult. Here is the Turbo Pascal version of scan, which, by the way, was tested with versions 1.00 and 2.00 of the Borland compiler running under PC-DOS 2.10 on an IBM XT.

```
FUNCTION scan(     limit  : integer;
                   opcode,
                   ch     : char;
              VAR source           ) : integer;
{Scan limit bytes of source looking for ch(opcode =)or not ch(opcode
{Return index of first true match.(0 = found at first byte of sourc
{Return limit if no match found. }
LABEL
   999;
TYPE
   char_array = ARRAY[0..maxint] OF char;
VAR
   a       : char_array absolute source;
   i       : integer;
   posscan : boolean;
BEGIN
   posscan := limit > 0;
   i := 0;
   WHILE i <> limit DO
   BEGIN
      CASE opcode OF
         '=' : IF a[i] = ch THEN GOTO 999;
         '#' : IF a[i] <>ch THEN GOTO 999
      END; { case }
      CASE posscan OF
         true:  i := succ(i);
         false: i := pred(i)
      END { case }
   END; { while }
999:   scan := i
END; { scan }
```

Notice that char_array is defined to be as large as possible by using the predefined constant, maxint. The maximum positive limit of an integer, then, is the maximum distance of a single

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

scan, a reasonable restriction. On the PC, this limits a scan of memory to 32,768 bytes (including index zero).

The declaration for array variable a may appear strange. By using the absolute identifier, we tell the Turbo compiler to assign the address of the untyped source parameter to this variable. Array a now starts at source. The absolute feature in Turbo Pascal provides a convenient way to address arbitrary variables anywhere in memory, while avoiding the use of pointers to specific data types.

You should be able to follow the body of the function with no further explanation. Index i simply increases or decreases until limit is reached, or the partial expression (using opcode and ch) is satisfied. This index is returned as the function value.

Purists may frown at the use of GOTOs to exit the WHILE loop, but after some experimentation with other approaches, I found the present form to be fast and concise. There is nothing wrong with GOTOs, of course, but I try to limit their use to quickly exiting loops in a downward direction. Engraved in stone is my rule to never use a GOTO to jump into another loop.

As an amusing exercise, try rewriting scan without the GOTOs. Harder, I believe, would be a GOTO-less scan that runs faster than the one printed here. (If you are in the mood for a challenge, I would be most happy to be proved wrong. No machine language solutions allowed. That would be cheating.)

HOW SCAN WORKS

The function value returned by scan is an integer offset, relative to the source of the scan, and within the range of limit, where the partial expression was first satisfied. This offset represents the location where ch was found (if using the equals opcode, '=') or where ch was not found (if using the not equals opcode, '#'). The offset might be positive if scanning forward (limit is positive), or negative if scanning backward (limit is negative).

If scan returns zero, then the partial expression was satisfied at the source, at the first byte scanned. This is true whether scanning forward or backward.

If scan returns the same value as the value of limit, then the partial expression was not satisfied within the range of bytes being scanned. This might signify that ch was not found anywhere within limit bytes of the source ('=' opcode), or that ch was found everywhere within the limit ('#' opcode). (Don't let the seemingly reversed logic fool you. When scanning for a nonoccurrence of something — for example, no nulls in a buffer — a failure tofind any "non-nulls" indicates that the buffer contains all nulls!)

In the interest of completeness, if limit is zero, then scan returns zero, and no scan is performed. A silly feature, perhaps, but consistent with the definition. (If scan equals limit, then the partial expression was not satisfied. Because no scan took place, the expression could never be satisfied.)

HOW TO USE SCAN

The most common use of scan is looking for particular characters in a text buffer. This might form part of the logic of a string search function in a word processing program. Another use is to scan a string looking for single character commands. B_command is a boolean variable in this example:

TallScreen™ TAKES THE DRUDGERY OUT OF DOS

- 1000 lines of screen output saved to scroll through for review and editing
- Commands and directory paths automatically saved for quick recall again and again
- Screen output and commands edited with standard, powerful, easy-to-learn and easy-to-use word processor techniques
- Screen output saved to file or printer, or frozen anywhere on the screen
- DOS commands help file included for quick reference
- Keyboard macros supported similar to the IBM Personal Editor.

For the IBM PC®, PCjr, XT, AT, or compatibles. Works with DOS 2.00 and above. Works with PROKEY™ and SIDEKICK™. Compatible with most mouse systems. Next day shipping on VISA/MC. MD & NM residents add sales tax.

$49.95
INTRODUCTORY PRICE

"TallScreen has completely changed the way I use DOS."
Bob Beilstein, PC user

"A godsend for power users."
Hanno Hinsch, Author of TELIOS

"This is the way DOS should have been designed in the first place!"
Rex Swain, Technical Director Parallax Systems

"Just recalling screen output and past commands alone is worth the price."
Edward T. Garner, VP Finance ARIES Group

For orders or information see your local dealer or call 800-522-0290

QUALITAS, INC., P.O. BOX 3AK/UPB
LAS CRUCES, NM 88003 505-522-0290

QUALITY PROGRAMMING TOOLS
QUALITAS

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
B_command := scan( length(s),'=','B', s[1] ) <> length(s);
```

B_command would be true only if the character "B" is found in the string s, which is subscripted [1] to start at the first character. (S[0] is reserved in Turbo and UCSD Pascal for the length of the string.) If B_command is false, then scan returned the value of limit, here equal to the length of the string. In that case, "B" was not found anywhere in the string. You may want to walk through what happens in this example if the length of the string is zero. The same example in UCSD Pascal would be written this way.

```
B_command := scan( length(s), ='B', s[1] ) <> length(s);
```

SCANNING BUFFERS

A neat "trick" is to use scan to return the number of characters in a buffer, assuming that buffers are always flushed with nulls after the last valid character, as is often the case.

```
PROGRAM buffers;
{ Example of scanning a character buffer }
CONST
 bufsize = 1023;      { number bytes in buffer minus one }
VAR
     b : packed ARRAY[ 0 .. bufsize ] OF char;
     n : integer;

BEGIN
 read_buffer( b );
 n := bufsize + scan(-bufsize,'#',chr(0), b[ bufsize ] );
 writeln( 'There are ', n, ' characters in the buffer.' )
END.
```

The read_buffer procedure is undefined. Integer n is set to the number of characters up to the first null in the buffer. This is accomplished by starting at the end of the buffer, at index = bufsize, and scanning backward for the first non-null. Verify for yourself that if n is zero after scanning, then the buffer is empty (contains all nulls). Remember that scan would return limit (-bufsize) if the partial expression is not satisfied.

A STRING STRIPPER

It is often necessary to remove, or strip, the leading and trailing spaces from a string. Although this can be done with FOR loops, a much speedier way can be devised with the scan function along with the procedure delete, which is identical in both the Turbo and UCSD Pascal systems.

```
IF length( s ) > 0 THEN
BEGIN
   s[0] :=
    chr( length( s ) +
         scan(-length( s ),'#',chr(blank),s[ length(s ) ] ));
   IF length( s ) > 0 THEN
         delete( s,1,scan( length( s ),'#',chr(blank),s[1] ))
END { if }
END; { strip }
PROCEDURE strip( VAR s : string255 );
{ Remove trailing and leading spaces from a string }
BEGIN
```

If your passed string is declared shorter than 255 characters, you'll have to surround the call to strip with the string checking option turned off:

```
{$v-} strip(s) {$v+}
```

Implementing scan in Turbo Pascal made converting much of my UCSD Pascal programming easier. I'm always finding new ways to use scan and am glad to have it in my Turbo library. □

YOUR CODE MAY BE WASTING ITS TIME!

THE PROFILER™ CAN HELP . . .

- **Statistical Execution Profiler**
- **Works with any language**
- **Completely configurable**
- **Up to 16 partitions in RAM/ROM**
- **Time critical code optimization**
- **Abnormal code behavior tracking**
- **Graphic presentation of results**
- **Easy to use menu interface**

THE PROFILER is a software package which gives you, the programmer, a powerful tool for locating time consuming functions in your code and allows you to performance tune your program. With **THE PROFILER** you can determine where to optimize your code for maximum benefit, then measure the results of your efforts.

Using **THE PROFILER**, you can answer questions like:
Where is my program spending its time?
Why is my program so slow? What is it doing?
Is my progam I/O bound? CPU bound? Are data buffers large enough?
How much improvement did my changes make?

THE PROFILER is completely software based and consists of a system resident driver and a monitor program. The memory partitions can range from 1 byte to 1 megabyte in size and can be anywhere in the address space.

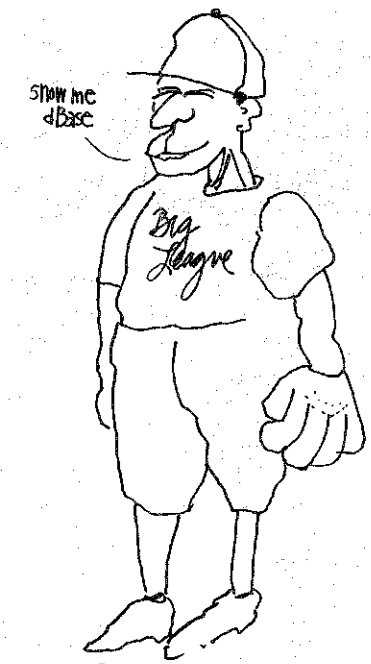
NO ADDITIONAL HARDWARE IS REQUIRED!

Requires an IBM PC or compatible system with a minimum 64k and one drive.

THE PROFILER is available for $125.00 **(New Low Price)** from DWB Associates or ask your software dealer. To order or for more information, call or write DWB Associates. VISA/MC accepted. Dealers welcome.

IBM is a trademark of IBM Corp. MSDOS is a trademark of Microsoft Corp.
THE PROFILER is a trademark of DWB Associates.

dwb
Associates
P.O. Box 5777
Beaverton, OR 97006
(503)629-9645



Famous Computer Personalities #210
Lefty Polowski, inventor of the dBase concept.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

CALENDAR DATE ROUTINES IN BASIC

by Bernard H. Robinson, Jr.

While there are calendar date routines published in various books, the books usually have copyright notices that prohibit one from copying the routines and using them in one's own programs. The routines presented in this article were written especially for publication here and do not use any copyrighted routines. They may be a little longer than some routines, but they have been carefully tested and work very well. These routines were developed from the basic principles of the calendar we use today, the Gregorian calendar, as described in standard references such as astronomy texts, encyclopedias, etc. The material in The World Almanac, published by Newspaper Enterprise Association,Inc., is especially recommended as a concise, but informative introduction. The Encyclopedia Britannica has some fascinating historical information. Your public library will likely have other references on the subject.

The routines presented here have the following capabilities and limitations:

1. They are for the modern calendar (Gregorian) in use in the U.S.A., Europe, and practically throughout the whole civilized world.

2. They are for the period September 14, 1752, through February 28, 4000, a span of almost 2200 years.

3. Given a calendar date, they will determine the day of the week for that date.

4. Given two dates, they will compute the number of days between the dates.

5. They will determine the date that is a specified number of days before or after a given date.

6. They will determine the Julian Day corresponding to a calendar date and the calendar date corresponding to a Julian Day.

Some of the terms used above may not be familiar to many readers. The history of calendars and the origins of the terms used are interesting topics, but space will allow us to discuss only the essential points here. The Gregorian calendar has been in use in this country since before the Revolutionary War. In the interest of simplicity, the programs presented here are for Gregorian calendar dates only. The range of dates over which they apply was chosen to begin on the date that the Gregorian calendar became the official calendar in Great Britain and her colonies (September 14, 1752). Dealing with dates before that date is complicated by the fact that some European countries adopted the Gregorian calendar as early as 1582, almost 200 years before Great Britain did. It is further complicated by the old practice of starting a new year on March 25 under the Julian calendar instead of the current practice of starting it on January 1. In some places, including in England and the American colonies, March 25 was New Year's Day until the Gregorian calendar was adopted; in other places, January 1 was made New Year's Day before the Gregorian calendar was adopted. Thus, rather detailed knowledge of which calendar and which new year's convention was in use in the place of interest is required for dates before September 14, 1752. Sometimes dates around the time of adoption of the Gregorian calendar are followed by the letters "o.s." for "old style" (Julian calendar) or "n.s." for "new style" (Gregorian calendar). Anyone needing to deal with dates from the old Julian calendar will probably find the information in the World Almanac helpful.

The choice of February 28, 4000, as the upper bound on the range of these routines is not entirely arbitrary. The algorithms used in the routines are good for Gregorian dates beyond this, but there are complications. The Gregorian calendar will be one day off from accurate astronomical time by the year 5000. There are various proposals for dealing with this, one of which is to make the year 4000 a common year instead of a leap year as is required by the Gregorian calendar rules. Since there is doubt whether the year 4000 will be a leap year and the utility of dates this far in the future is of doubtful value anyway, the day before leap day in the year 4000 seemed to be a good point to end the range of the routines.

The term "Julian Day" is only indirectly related to days on the old Julian calendar. The Julian calendar was named after Julius Caesar, who put it into effect. The Julian Day system was devised by an astronomer, Joseph Scaliger, in 1582, who named the system after his father, Julius. The Julian Day system numbers days serially starting with January 1, 4713 B.C. (Julian calendar) and has been used mainly by astronomers. As astronomers use it, the Julian Day corresponding to a given date starts at noon on the date, but that is of no consequence to anyone but astronomers. The idea of serially numbering days from some convenient starting point is useful in computing the number of days between dates. In the DAYSUB.BAS routines, the Julian Day is used instead of an arbitrary serial number. This was done partly as a convenience in checking the accuracy of the routines, since the results obtained with them could then be compared with the results of other programs that use the Julian Day system.

The program DAYSUB.BAS starts with an initialization routine, lines 500 through 650. This routine provides a dimension

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

statement for arrays, loads arrays with data, defines some other constants, and defines three special functions in DEF FN statements. This initialization routine needs to be executed before the rest of the program, but only needs to be executed once while running the program.

Aside from this initialization routine, three subroutines constitute the heart of the program. The subroutine starting at line 2000 computes the Julian Day corresponding to a given date on the Gregorian calendar. It must be entered with the variables n.month% set equal to the number of the month (1 for January, 2 for February, etc.), day% set equal to the day of the month, and year% set equal to the year in full (1984, not just 84). The bulk of the 2000 subroutine consists of statements that check whether the input date is a valid date. This is necessary since a date such as November 31 in any year or February 29 in a year that is not a leap year would otherwise yield a response that looks meaningful but is not. The 2000 subroutine also verifies that the date is in the range of the program. Once the date is determined to be valid, execution of line 2150 determines both the Julian Day and the day of the week of the input date. The day of the week is obtained by adding one to the Julian Day and determining the remainder when the sum is divided by 7. The remainder is the number of the day of the week (0 = Sunday, etc.).The BASIC MOD function is not used here because the function does not work on double precision numbers.

The short subroutine at line 3200 determines whether the input date is in a leap year. This routine is called in both subroutine 2000 and in subroutine 5000.

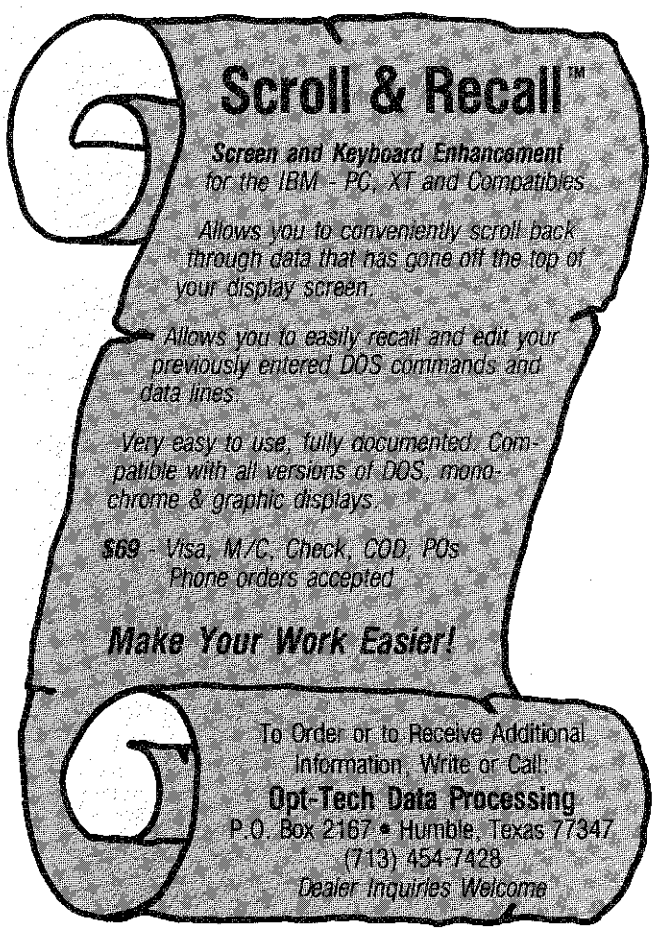
The subroutine starting at line 5000 must be entered with JD3# set equal to a Julian Day number within the range of the routines. The subroutine returns with n.month%, date%, and year% set equal to the calendar date corresponding to the input Julian Day.

The lines from 1000 to 1800 are not really part of the routines themselves, but are included to show how to use the routines. This part of DAYSUB.BAS has been kept short and simple in an attempt to make this demonstration as clear as possible. First comes a simple selection routine to allow the operator to select which function to demonstrate. A one-digit entry without the enter key is all that is necessary. Once the desired routine is selected, the input of the necessary data is requested. Dates are entered in numerical form. Note that commas must be used as separators between the month number and the day number and between the day number and the year number. At the conclusion of each option, the results are printed to the screen and the program pauses to let the operator see the results. Typing any key will then send the program back to the option selection routine.

If Option 1 is selected, a single input date is requested. As soon as the Julian Day and day of the week have been computed to demonstrate the 2000 subroutine, the program continues, using the Julian Day just computed as the input for the 5000 subroutine, which then computes the date corresponding to the Julian Day. Thus for the Option 1 demonstration, the date returned by the 5000 subroutine must always be the same as the date entered by the operator originally. The demonstration program was designed this way to avoid having to enter a long (seven-digit) Julian Day number just for a demonstration. It also simplifies verification that the 5000 subroutine is working correctly.

Option 2 computes the number of days between two input dates. When Option 2 is selected, the program requests entry of the first date. It uses subroutine 2000 to compute the Julian Day of this entry and stores it in JD1#. The program then requests entry of the second date and computes the Julian Day for it. The Julian Day of the first date is then subtracted from that of the second date, yielding the number of days between the dates, which is then printed.

Option 3 determines the date at a specified number of days before or after a given date. When this option is selected, the program requests entry of the given date and then the number of days before or after the date. The program computes the


Scroll & Recall™
Screen and Keyboard Enhancement
for the IBM - PC, XT and Compatibles
Allows you to conveniently scroll back through data that has gone off the top of your display screen.
Allows you to easily recall and edit your previously entered DOS commands and data lines.
Very easy to use, fully documented. Compatible with all versions of DOS, monochrome & graphic displays.
$69 - Visa, M/C, Check, COD, POs
Phone orders accepted
Make Your Work Easier!
To Order or to Receive Additional Information, Write or Call:
Opt-Tech Data Processing
P.O. Box 2167 • Humble, Texas 77347
(713) 454-7428
Dealer Inquiries Welcome

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Julian Day for the given date, adds the days difference to the Julian Day, then uses subroutine 5000 to convert the new Julian Day to its corresponding calendar date. If the date and number of days difference entered would result in a date out of the range of the program, the program detects this and notifies the operator that the input information is out of range.

The DAYSUB.BAS routines are based on the following rules of the Gregorian calendar: The calendar is made up of common years and leap years. The common years have 365 days in the familiar 12 months. Leap years have 366 days, the extra day being added as February 29. With certain exceptions, any year that is exactly divisible by 4 is a leap year. The exceptions are covered by the following rule: Any year that ends in "00" such as 1900 is not a leap year unless the year is exactly divisible by 400.

The calendar date to Julian Day routine is fundamental to the whole program. The following is a brief outline of the considerations that led to this implementation. Some of the points involved are not easy to see and can be understood only by carefully thinking them through. A reference date (December 31, 1751) is used in the calculations and it was chosen as the last day of the year proceeding the first year in the range of the program. The Julian Day for this reference date is 2,360,964, which is the meaning of the constant JB# defined in line 560.

The defined function of line 610 performs most of the calendar date to Julian Day calculations. The term (365*(year%-1752)) yields a first approximation of the number of days from the reference date to the end of the year just prior to that of the given date. This approximation must be corrected for leap years. The term int((year%-1749)/4) gives a first approximation of the number of leap days that must be added as a correction to the (365*(year%-1752)) term, but this term must be corrected for the exceptions rule (see above). The defined function of line 590 computes the number of days to add to compensate for the exceptions to the basic leap year rule. These steps yield the correct number of days from December 31, 1751, through December 31 of the year just previous to the given year.

Next, the number of days into the given year must be determined. This is done with the aid of the num.d% array. This array stores the number of days from January 1 to the beginning of each month of the year for common years and for leap years. This array has two dimensions, the second index of which is set to 0 for regular years and to 1 for leap years. As a specific example, num.d%(3,0) contains the number of days from the beginning of the year to the beginning of March for regular years and num.d%(3,1), that for March in leap years. (The num.d%(0,0) and num.d%(0,1) terms are not used in this routine. They contain the total number of days in regular years and leap years; these are used in the 5000 subroutine.) With the num.d% factor added in, we have the number of days up to the beginning of the month of the input date. Finally, we add day%, the number of the day of the month, and now we have the total number of days from December 31, 1751. If we add to this JB#, the Julian Day of December 31, 1751, the result is the Julian Day of the given date.

The routines use logical expressions in several places. For example, in line 3210

(YEAR% MOD 100 = 0)

is a logical expression that evaluates to TRUE when the year of the input date ends in "00". The purpose of line 3210 is to branch to line 3230 when the year does not end in "00". Similar logical expressions are used in lines 3220 and 3230. To understand the use of some of the logical functions in the program, recall that a logical TRUE is numerically equal to -1 and a logical FALSE is equal to 0.

A few words of caution are in order. If you want to use this program, be sure that it is entered exactly as it is given here. Pay special attention to including the type characters (# for double precision and % for integer) on variables and constants. These were included for specific reasons and must be

THE HAMMER

Software Tools in C

Designed for the programmer with the user in mind.

More than just BIOS/DOS access. , THE HAMMER ALSO provides screen attribute control, Smartmodem control, UNIX-like pattern matching, 123-like command menus, date & time processing routines, easy data entry & verification of dates, numbers, and strings, AND MORE. Super data entry routines give the programmer an easy to use, solid interface with the user.
Requires: IBM-PC/XT, DOS2, CI-C86 Compiler
Other C compilers supported soon.

Routines tried & true. $110 with source and manual.
For ordering or inquiring, call or write now:

O E S SYSTEMS
1906 Brushcliff Road
Pittsburgh, PA 15221
412/243-7365

Looking for the right tool for the job?
GET THE HAMMER

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

inserted. Remember also that BASIC considers a variable such as "day" different from the variables "day%" or "day#". The accidental omission of these integer precision characters can result in erroneous results from the calculations. One of the best ways to find this kind of error quickly is to use a good cross reference program.

Another caution: A letter in the User-to-User column in the October 2, 1984, PC Magazine states that a very popular spreadsheet program has an error in its routine to convert dates to date serial numbers (which it uses in a way similar to the way the DAYSUB.BAS routines use Julian Days) . The letter writer notes the routine considers the year 1900 to be a leap year and 2000 not to be a leap year, which is just the reverse of the true situation. Thus that program makes errors if one date is between March 1, 1900 and February 28, 2000 and the other is not.

In the course of verifying the accuracy of the DAYSUB.BAS routines, I compared the results with the results of the program "JULDY" in the book "Celestial BASIC", by Eric Burgess, published by Sybex, Inc. I have found that the "JULDY" program is not correct for all dates; it gives Julian Day numbers that are off by one day for dates from January 1 through February 28 for most years that end in "00", such as 1800 and 1900. It does give correct results for the year 2000. Because of its errors, it gives the same Julian Day number for December 31 of the previous year ending in "99" and for January 1 of the year ending in "00" when the year ending in "00" has the error. Be warned, then, that if you compare the results of this program with those of any other program and do not get the same answers, do not conclude that this program is at fault without further investigation.

The printed results from the DAYSUB.BAS program given in this article have been carefully checked against other sources. These sources include manual methods published in The World Almanac and comparison with other programs. Conversions between Julian Days and calendar dates were verified by comparison with results obtained with a 10-digit accuracy programmable calculator (H.P.41-CV) using a calendar program in a ROM developed and sold by a very reliable H. P. users group (the PPC). The range of the program in the PPC ROM extends considerably beyond the range of the routines in DAYSUB.BAS. A table of examples of results obtained with the DAYSUB.BAS routines is provided. The table includes the input date, the Julian Day number, the day of the week, and the results of converting the Julian Day back to a calendar date. You should verify that your program gives the same results for all of the examples given here to insure that you have made no typographical errors in entering the program into your computer. The program functions on either BASIC or BASICA and on either DOS 1.1 or 2.0.

One final word about Julian Days. Apparently Joseph Scaliger intended that the Julian Day numbering system should be recycled after a period of 7980 years (Julian calendar). The first cycle will be completed on December 31, 3267 (Julian), and the new cycle should start with Julian Day = 0 on January 1, 3268 (Julian), or January 23, 3268, on our Gregorian calendar. This date is within the range of the routines, but the routines do not recycle the Julian Day to zero; the Julian Days just keep on increasing to facilitate their use in determining days between dates. If the astronomers do indeed recycle the Julian Day system and if any readers are still using the routines when the recycle occurs, I shall be happy to send them a patch to the programs upon receipt of a self-addressed, stamped envelope.

In addition to doing practical business date calculations, the routines will do fun things like tell you how many days you have lived so far, how many days you have been married, how many days since the Declaration of Independence was signed, etc.

Have fun!

Calendar Date Routines
B. H. Robinson
146 Homestead Ave.
DeBary, FL 32713

$35 DISK CACHE/PRINT BUFFER

Disk Cache
-Look ahead background disk reads
-Background disk writes
-Automatic detection of disk removal

Print Buffer
-Use inexpensive computer RAM to replace expensive hardware print buffer
-Smooths background printing

Plus
-User specified buffer size (10K - 600K)
-Dynamic partitioning of buffer to the disk cache and print buffer.

$25 DEBUG ENHANCER
-Set breakpoint at current instruction
-Use register names in command line
-Attach data line to register display
-One-key trace for int, call, rep & loop
-Expanded Hex Command incl. conversions

To order call (503) 883-3278
or write OREGON PROGRESSIVE SOFTWARE
1965 Manzanita Street
Klamath Falls, OR 97601

VISA or **MASTER CARD** Accepted. Mailed orders must indicate DOS version number and machine type. We pay domestic shipping and handling.Extra charge for C.O.D. and orders outside the continental U.S.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
10 ' Program name: DAYSUB.BAS
20 '   Written by: B. H. Robinson
30 '      Address: 146 Homestead Ave., DeBary, FL 32713
40 '         Date: September 24, 1984
50 '      Version: 018
500 ' Dimension and initialization routine
510 DIM NUM.D%(12,1),DAY.WK$(6)
520 DATA 365,0,31,59,90,120,151,181,212,243,273,304,334
530 DATA 366,0,31,60,91,121,152,182,213,244,274,305,335
540 FOR K1%=0 TO 1 : FOR I1%=0 TO 12
550 READ NUM.D%(I1%,K1%) : NEXT I1%,K1%
560 JB#=2360964#
570 TRUE%=-1 : FALSE%=0
580 VALID%=TRUE% : AE$=""
590 DEF FNCOMP.CENT%=INT((YEAR%-1601)/400)-INT((YEAR%-1701)/100)
600 DEF FND.PTR%(D1#)=INT(D1#+1-INT((D1#+1)/7#)*7#+.0000001#)
610 DEF FNDAYS#=FNCOMP.CENT%+NUM.D%(N.MONTH%,L.YEAR.FLG%*-1)
                +DAY%+INT((YEAR%-1749)/4)+(365*(YEAR%-1752))
620 DATA "SUN","MON","TUE","WED","THU","FRI","SAT"
630 FOR I%=0 TO 6 : READ DAY.WK$(I%) : NEXT I%
640 '
650 ' end of initialization **********************
660 '
1000 ' Examples of using the routines
1010 CLS
1020 PRINT "     Select desired routine by typing 1, 2, 3, or 4"
1030 PRINT " 1  for date to day of week and Julian Day routine."
1040 PRINT " 2  for number of days between given dates routine."
1050 PRINT " 3  for Date a given number of days from a date."
1060 PRINT " 4  to end program at this point."
1070 PRINT
1080 INK$=INKEY$ : IF INK$="" THEN 1080
1090 IF VAL(INK$)<1 OR VAL(INK$)>4 THEN BEEP : GOTO 1080
1100 IF INK$="4" THEN END ELSE ON VAL(INK$) GOTO 1110,1240,1330
1110 ' Example of date to Julian day conversion follows:
1120 INPUT "Enter Month,day,year: ",N.MONTH%,DAY%,YEAR%
1130 GOSUB 2000 ' compute Julian day
1140 IF VALID% THEN 1150 ELSE 1800
1150 PRINT "Julian day: ";D1#,"Day of week: ";DAY$
1160 '
1170 ' Example Julian day to date conversion
1180 JD3#=D1#
1190 GOSUB 5000
1200 IF VALID% THEN 1210 ELSE 1800
1210 PRINT "Julian Day reconverted to date is:"
1220 PRINT "Month: ";N.MONTH%,"Day: ";DAY%,"Year: ";YEAR%
1230 GOTO 1430
1240 ' Example of days between dates computation:
1250 INPUT "Enter first date: month,day,year: ",N.MONTH%,DAY%,YEAR%
1260 GOSUB 2000 : IF VALID% THEN 1270 ELSE 1800
1270 JD1#=D1#
1280 INPUT "Enter second date: month,day,year: ",N.MONTH%,DAY%,YEAR%
1290 GOSUB 2000 : IF VALID% THEN 1300 ELSE 1800
1300 DIF.DAYS#=D1#-JD1#
1310 PRINT "Difference = ";DIF.DAYS#;" days."
1320 GOTO 1430
1330 ' Example of Computing a date x no. of days from a given date
1340 INPUT "Enter Starting Date: Month,day,year: ",N.MONTH%,DAY%,YEAR%
1350 PRINT "Enter Number of days before (-) or after (+) date."
1360 INPUT D.DAYS#
1370 GOSUB 2000 ' compute Julian day of first day
1380 IF NOT VALID% THEN 1800
1390 JD3#=D1#+D.DAYS# ' J. D. of new date
1400 GOSUB 5000 ' Convert Julian day to date
1410 IF NOT VALID% THEN 1800
1420 PRINT "New Day is: ";N.MONTH%;"/";DAY%;"/";YEAR%
1430 PRINT : PRINT "Type any key to continue"
1440 INK$=INKEY$ : IF INK$="" THEN 1440 ELSE 1000
1800 PRINT AE$ : PRINT "BAD DATA" : GOTO 1020
1810 '
1820 '
2000 ' Verify input date is valid and compute Julian day
2010 ' Enter with n.month%=month, day%=day, and year%=year
2020 ' Returns with d1# = Julian day and day$ = day of week
2030 VALID%=TRUE% : IF N.MONTH%<1 OR N.MONTH%>12 THEN AE$="Bad month"
                    : GOTO 2140
2040 IF DAY%<1 OR DAY%>31 THEN AE$="Bad day" : GOTO 2140
2050 TEST.D#=DAY%+N.MONTH%*100#+YEAR%*10000#
2060 IF TEST.D#>17520913# AND TEST.D#<40000229# THEN 2080
2070 AE$="Date out of range." : GOTO 2140 ' Bad date, exit
2080 GOSUB 3200 ' Set L.YEAR.TAG%=TRUE% if year is leap year
2090 IF DAY%<29 THEN 2150 ' date is valid for any month
2100 ON N.MONTH% GOTO 2150,2120,2150,2110,2150,2110,2150,2150,
                                  2110,2150,2110,2150
2110 IF DAY%<31 THEN 2150 ELSE AE$="Thirty day month." : GOTO 2140
2120 IF DAY%>29 THEN 2130 ELSE IF L.YEAR.FLG% THEN 2150
                                  ELSE AE$="Not leap year." : GOTO 2140
2130 AE$="Date not valid in Feb." : GOTO 2140
2140 VALID%=FALSE% : BEEP : GOTO 2160
2150 D1#=FNDAYS#+JB# : DAY$=DAY.WK$(FND.PTR%(D1#))
2160 RETURN
3200 ' Set l.year.flg%
3210 IF NOT (YEAR% MOD 100 = 0) THEN 3230
3220 L.YEAR.FLG%=(YEAR% MOD 400 = 0) : GOTO 3240
3230 L.YEAR.FLG%=(YEAR% MOD 4 = 0)
3240 RETURN
5000 ' Julian date to Cal. date routine
5010 ' Enter with JD3# = to a Julian Day
5020 ' Returns with n.month%, day%, and year% = corresponding date
5030 DIF.DAYS#=JD3#-JB#
5040 E.YEAR%=1752 + INT(DIF.DAYS#/365.2422)
5050 DAY%=31 : N.MONTH%=12 : YEAR%=E.YEAR%-1
5060 IF YEAR%<1751 OR YEAR%>4000 THEN 5100
5070 GOSUB 3200 ' set l.year.flg% for E.YEAR%-1
5080 D1#=FNDAYS#+JB# : DAY$=DAY.WK$(FND.PTR%(D1#))
5090 GOTO 5120
5100 VALID%=FALSE% : AE$="DATE OUT OF RANGE"
5110 BEEP : GOTO 5240
5120 DIF.DAYS2#=JD3#-D1#
5130 IF DIF.DAYS2#=0 THEN 5240
5140 IF DIF.DAYS2#<0 THEN E.YEAR%=E.YEAR%-1 : GOTO 5050
5150 YEAR%=E.YEAR% : GOSUB 3200 ' Set l.year.flag% for E.YEAR%
5160 K1%=-1*L.YEAR.FLG% ' k1%=1 if leap year, 0 if not
5170 IF DIF.DAYS2#>NUM.D%(0,K1%) THEN E.YEAR%=E.YEAR%+1 : GOTO 5050
5180 FOR I1%=12 TO 1 STEP -1
5190 IF NUM.D%(I1%,K1%)<DIF.DAYS2# THEN N.MONTH%=I1% : I1%=1
5200 NEXT I1%
5210 DAY%=DIF.DAYS2#-NUM.D%(N.MONTH%,K1%)
5220 TEST.D2#=DAY%+N.MONTH%*100#+YEAR%*10000#
5230 IF TEST.D2#<17520914# OR TEST.D2#>40000228# THEN 5100
5240 RETURN
```

□

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

CALCULATE RECORD LAYOUTS, AUTOMATICALLY !!

by J. Patrick Foley

Spreadsheets are a great tool for financial planning, forecasting and other numerically oriented tasks. Though not widely considered one of the standard tools, spreadsheets can be a powerful tool for the software developer. This article will illustrate one method of tapping the power of the electronic spreadsheet to document and calculate record descriptions — one of the critical steps in the design of any software system.

The record description defines many of the specifics about a record and the fields it contains. Though not meant to be a definitive list, the following contains the items most often contained in a record description:

* A general description of the file and its contents.
* The file name as it appears on the directory.
* The record length of the file.
* The access method.
* Names of key fields.
* For each field, the following items:
    * The field name.
    * A description of the field contents.
    * The format of data within the field.
    * The number of bytes occupied by the field.
    * The beginning and ending byte locations.
    * Other comments

The record description used as an illustration will include each of these specifications.

The advantages of adapting the record description to a spreadsheet are many. The major advantage is the recalculation of field lengths and field offsets that occurs whenever a field size is changed or a new field is added to a record. These changes occur frequently throughout the development phase of a system, and more often than realized during the life cycle of a computerized system. The number of available bytes and starting and ending record locations are recalculated automatically with a spreadsheet.

Another advantage is the computerization of the design process. All too often, software designers will produce a record description and never update it. By entering the record description into a spreadsheet, it can be updated and reprinted in a matter of minutes. No more handwritten updates to the only typewritten page in the documentation manual!

A third advantage is the maintenance of a data dictionary. A data dictionary can be defined as a database of data fields and characteristics of the fields. Using the extraction and file export functions of spreadsheets allows the designer to build a list of data fields and all pertinent information about each field.

Once the record description has been digitized into spreadsheet form, the designer can build utilities to make his or her job even easier. For example, a utility that automatically generates the file access programming code can be written. As an illustration, the utility could create "include" files (record layouts) that contain the FIELD statements from the name and byte count columns of the spreadsheets.

SPREADSHEET STRUCTURE

The file that is used in the example will be a customer master file. Fields contained in the master file include the customer number, name, address (3 lines), city, state, zip, phone, person to contact, comments, last activity date and extra space for future needs. Refer to Figure 2.

The spreadsheet is composed of two main sections - the header and detail sections. The header section describes attributes of the file being defined. The attributes included in the example spreadsheet are the following:

FILE DESCRIPTION ...	A general description of the file and its usage.
DOS FILE NAME	The file name used to access the file.
PRIMARY KEY	The name of the field(s) that are record keys for indexed access methods.
RECORD LENGTH	The logical record length.
ACCESS METHOD	An indication of the type of file access allowed against the file.
LAST UPDATED ON ...	The date that this spread sheet was last updated.

The detail section provides pertinent information about each field in the record. The spreadsheet columns contain the corrosponding record format data according to the following table:

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

COLUMN	CONTENTS	DESCRIPTION
A	Field name	The name assigned to the field at the program level.
C	Description	A description of the usage of the field.
E	Format	An indication of the type of data, where A=alpha, N=numeric, X=any, (nn) for repetition.
G	Size	The number of bytes used by the field.
I	From	The starting byte position of the field.
K	To	The ending byte position of the field.
M	Comments	General comments pertinent to users of the record format.

BUILDING AND USING THE SPREADSHEET

The spreadsheet used for this illustration is Lotus 123, Version 1A. The spreadsheet will be built in two steps. First, a skeleton spreadsheet will be entered, along with certain default values, and saved. Figure 1 shows the completed skeleton spreadsheet. The instructions for building the skeleton spreadsheet appear in Instruction Set 1.

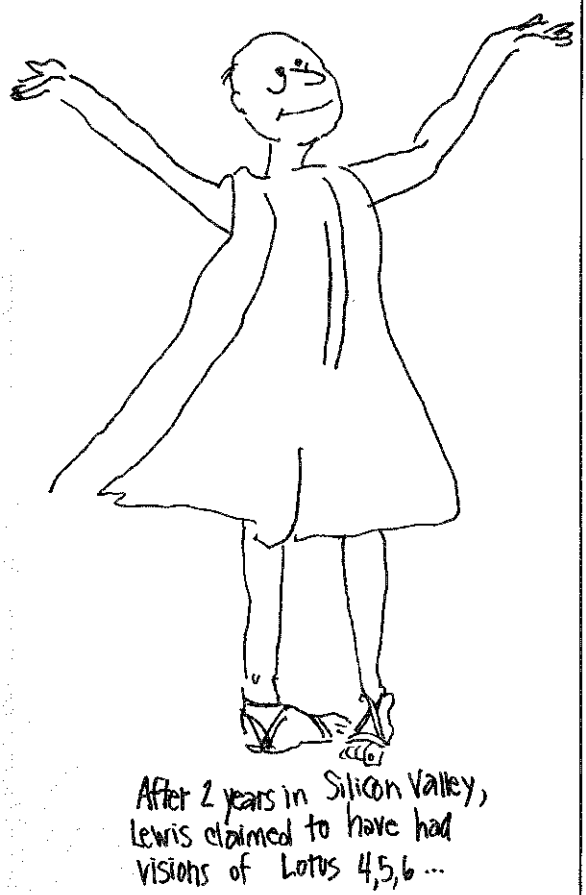
To build the customer record application spreadsheet, follow the instructions in Instruction Set 2. The completed customer worksheet appears in Figure 2.

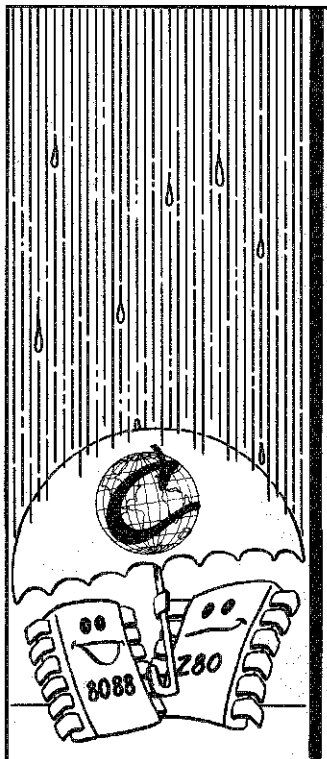
Now the fun begins. The spreadsheet from Figure 2 is now in the computer's memory. Notice that there are 84 available bytes at the end of the record. This is far too many for predictable future needs. Let's reduce the record length to 250. Lotus recomputes the available space to be 14 in a New York nanosecond.

Suppose that historical data shows that the first address line will require 30 positions. Instantly, the record offsets are recalculated. The available space has been reduced to 4 bytes.

Some more fine tuning reveals that only 15 characters are needed for the phone number. Once again, Lotus responds with 9 unused bytes.

Record layout spreadsheets is only one application available to the software designer. There are many other tedious tasks that can be alleviated with clever spreadsheet design. Analyze the way you work. You may surprise yourself with the number of ways your productivity can be increased by automating some of the design drudgeries of our profession.





An Optimizing, hassle-free C Compiler for the 8086-8088.

Ecosoft's Eco-C, the performance leader among full C compilers for the Z80, is now available for the 8086-8088 running under MSDOS (2.0 or later). Eco-C has features not found in any other 8086-8088 C compiler.

★ Over 100 library functions. Since they follow UNIX standards, your programs are highly portable in "both" directions ("up" to a UNIX machine or "down" to our Z80 compiler). This means new markets for your software at minimum development cost.
★ A single library. No more "dual" libraries and trying to remember what has to be linked with what. Your programs automatically take advantage of an 8087 if one is present at runtime.
★ A single floating point answer. No more "fuzzy floating point": your programs produce the same answers whether the floating point is done in hardware (8087) or software.
★ Error messages in English. No more cryptic numbers to look up. We tell you where the error occurred, what was found there, and what should have been there.
★ Strict syntax parsing. LINT is going to uncover fewer surprises because our parser looks hard at the details.
★ Efficient code. The optimizer pass of the compiler generates assembler code in Intel mnemonics that rivals that produced by compilers costing twice as much.

The price of the Eco-C compiler is $250.00 (all versions), including the user's manual, and is designed for use with Microsoft's MASM (or compatible) assembler and linker. When ordering, please specify disk format and whether you want the Z80-CP/M or 8088-MSDOS version of Eco-C.

ECOSOFT INC.

Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220
(317) 255-6476

MasterCard VISA AMERICAN EXPRESS

Eco-C (Ecosoft), MSDOS (Microsoft), UNIX (Bell Labs), CP/M (Digital Research), Z80 (Zilog), 8086, 8087, 8088 (Intel).

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
--- INSTRUCTION  SET  1  ---

---  BUILD SKELETON SPREADSHEET  --
```

STEP	DESCRIPTION	CELL	COMMAND
1	Set standard numerical format for entire spreadsheet (no decimal places).	A1	/WFFF0<enter>
2	Set standard column widths to 2.	A1	/WGC2<enter>
3	Set column widths for spreadsheet text and data columns.	A1 C1 E1 G1 I1 J1 K1 M1	/WCS10<enter> /WCS25<enter> /WCS10<enter> /WCS5<enter> /WCS5<enter> /WCS1<enter> /WCS5<enter> /WCS35<enter>
4	Enter header line.	C1	' RECORD DESCRIPTION (10 spaces preceeding)
5	Enter header information.	C3 C4 C5 C6 C7 C8	'FILE DESCRIPTION: 'DOS FILE NAME: 'PRIMARY KEY: 'RECORD LENGTH: 'ACCESS METHOD: 'LAST UPDATED ON:
6	Enter default record size.	E6	64
7	Enter default access method.	E7	'RANDOM
8	Set update date to current date, using day-month-year format.	E8 E8	@TODAY<enter> /RFD1<enter>
9	Enter column titles.	A11 C11 E11 G11 I11 M11	'FIELD NAME 'FIELD DESCRIPTION 'FORMAT 'SIZE 'FROM - TO 'COMMENTS
10	Enter dashed line for column separators.	A12 C12 E12 G12 I12 J12 K12 M12	\- \- \- \- \- '- \- \-
11	Enter first detail line of skeleton.	A13 C13 E13 G13	'FLD-01 'Field 1 'X(5) 5
12	Enter next detail line of skeleton.	A14 C14 E14 G14	'FLD-xx 'Field xx 'X(5) 5
13	Enter last detail line of skeleton, assuming that available space is reserved at end of record.	A15 C15 E15	'AVAILABLE 'Available Space 'X(nn), nn =
14	Enter starting position of first field.	I13	1
15	Enter formula for computing ending position of each field.	K13	+G13+I13-1
16	Enter formula for computing starting position of subsequent fields.	I14	+K13+1
17	Copy formula for computing ending position.	K14	/CK13<enter><enter>
18	Enter dashes in FROM/TO detail lines.	J13 J14	'- '-
19	Compute available space.	G15	+E6-K14<enter>
20	Copy formula for computing beginning and ending position.	I15	/CI14.K14<enter><enter>
21	Save skeleton worksheet as RD-SKEL.	G14	/FSRDSKEL<enter>

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

--- INSTRUCTION SET 2 ---

--- BUILD APPLICATION SPREADSHEET ---

STEP	DESCRIPTION	CELL	COMMAND
1	Load skeleton spreadsheet into memory.	A1	/FRRDSKEL<enter>
2	Enter header info - Description	E3	'Customer Master File
	" DOS File Name	E4	'CUSTMAST.DAT
	" Key Field Name	E5	'CUSNUM
	" Record Length	E6	320
	" Access Method	E7	'Random Indexed
3	Open up blank detail lines. The number of lines required is equal to the total number of fields - 2. In this example, there are 11 lines, extending from cell A14 to A25.	A15	/WIRA14.A24<enter>
4	Copy skeleton detail of FLD-xx 11 times.	A15	/CA14.M14<enter>A15.A24<enter>
5	Enter data for fields according to the following table:		

Col =	A	C	E	G	M
Row	Name	Description	Format	Size	Comments
13	CUSNAM	Customer Name	X(40)	40	Last, First and Middle
14	CUSNUM	Customer Number	N(7)	7	
15	ADDR1	Address Line 1	X(20)	20	
16	ADDR2	Address Line 2	X(20)	20	
17	ADDR3	Address Line 2	X(20)	20	
18	CITY	City	X(20)	20	
19	STATE	State Code	XX	2	
20	ZIP	Zip/Postal Zone	X(11)	11	
21	PHONE	Phone Number	X(20)	20	
22	CONTACT	Person to Contact	X(30)	30	
23	COMMENT	General Comments	X(40)	40	
24	DLACTV	Date of Last Activity	N(6)	6	

6	Correct AVAILABLE SPACE formulas.	G25	+E6-K24
		I25	/CI14.K24<enter><enter>
7	Save the spreadsheet as RDCUST	I25	/FSRDCUST<enter>
8	Print completed worksheet:	I25	/PP
	Set cell range to be printed		RA1.M25<enter>
	Set to compressed print (for EPSON)		OS\015<enter>Q
	Print it		G
	Exit Print Command		Q

-- FIGURE 1 SKELETON SPREADSHEET --

RECORD DESCRIPTION

FILE DESCRIPTION:
DOS FILE NAME:
PRIMARY KEY:
RECORD LENGTH: 64
ACCESS METHOD: RANDOM
LAST UPDATED ON: 24-Sep-84

FIELD NAME	FIELD DESCRIPTION	FORMAT	SIZE	FROM - TO	COMMENTS
FLD-01	Field 1	X(5)	5	1 - 5	
FLD-xx	Field xx	X(5)	5	6 - 10	
AVAILABLE	Available Space	X(nn), nn =	54	11 - 64	

-- FIGURE 1 SKELETON SPREADSHEET --

-- FIGURE 2 APPLICATION SPREADSHEET --

RECORD DESCRIPTION

FILE DESCRIPTION: Customer Master File
DOS FILE NAME: CUSTMAST.DAT
PRIMARY KEY: CUSNUM
RECORD LENGTH: 320
ACCESS METHOD: Random Indexed
LAST UPDATED ON: 08-Oct-84

FIELD NAME	FIELD DESCRIPTION	FORMAT	SIZE	FROM - TO	COMMENTS
CUSNAM	Customer Name	X(40)	40	1 - 40	Last, First
CUSNUM	Customer Number	9(7)	7	41 - 47	and Middle
ADDR1	Address Line 1	X(20)	20	48 - 67	
ADDR2	Address Line 2	X(20)	20	68 - 87	
ADDR3	Address Line 3	X(20)	20	88 - 107	
CITY	City	X(20)	20	108 - 127	
STATE	State	XX	2	128 - 129	
ZIP	Zip/Postal Zone	X(11)	11	130 - 140	
PHONE	Phone Number	X(20)	20	141 - 160	
CONTACT	Person to Contact	X(30)	30	161 - 190	
COMMENT	General Comments	X(40)	40	191 - 230	
DLACTV	Date of Last Activity	9(6)	6	231 - 236	
AVAILABLE	Available Space	X(nn), nn =	84	237 - 320	

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ONE PROGRAMMER'S OPINION

by Bill Salkin, Copyright (C) 1984 by ABComputing

If you have high blood pressure, I implore you to stop reading this article right now, for I can't be held responsible for what may happen to you! While I won't swear, my remarks will be nonetheless inflammatory. In a nutshell, I attack the cherished high-level languages that programmers enjoy using and, after bruising a few egos, attempt to convince you that, as a programmer, you have only one real recourse:

Program in Assembly Language and wait until Ada becomes a reality.

Gasp! Has the boy gone mad? (Someone in the audience yells, 'Get him a drink!") Now that I have my boxing gloves on, let's jump into the ring and battle the contenders, one by one.

BASIC

You've got to be kidding! That language has been given more face-lifts than a movie star. Structured preprocessors, tummy tucks, programs to remove line numbers, comments, etc. Even the authors of BASIC realize that it is time to rewrite their child. Let's let the old gray mare go out to pasture and live its remaining days in peace.

BASIC does have a reasonably useful string- handling capability, and it is highly interactive. But other languages are catching up and, with tear in eye, we must bid farewell to the reigning champ. (Funeral services, yet to be announced.)

C

A good language for creating C-monsters. Programmers tend to believe that C is a high-level assembly language and, accordingly, write code that even surpasses assembly language in mystery and suspense.

C programmers try to write very tiny functions, often performing only one easy-to-document service. Unfortunately, the time required to call one of these small functions may exceed the time required to perform the service. Not too good for performance.

C has a high degree of standardization, or at least I hope it does. The enforcement mechanism is programmer discipline, which has not always been a sure-fire solution. Standardization is a valuable feature, for it guarantees that a language, with slight modification, will run on other computer systems.

C structures are strictly for those wishing to scratch their heads for a long time, and for those relishing unbalanced structures. "If's" with no "end if's," "switch" statements needing "break" statements to be useful — c'mon guys, let's get real.

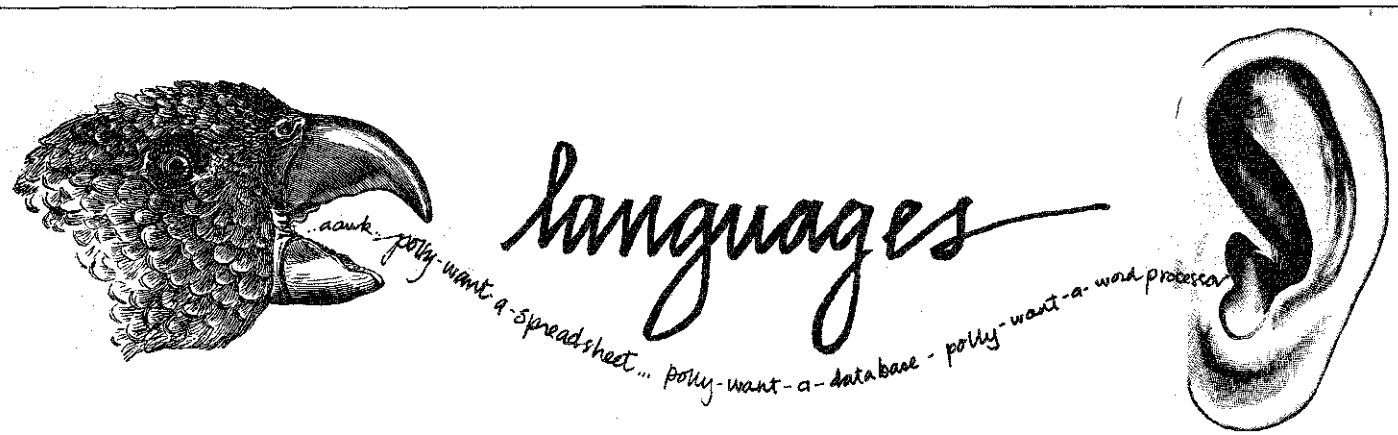
In short, not a bad language, just one that has gone astray and created C-monsters.

FORTRAN

Useful for numerical calculations. Not very good at other things. My favorite example is PT(1) = 0, PT(2) = 1; well, I actually meant PT(RED) = GREEN, PT(BLUE) = YELLOW. Sigh. Newer FORTRANs are essentially vitamin-enriched versions of older ones. You can pile on the makeup, but the same person is buried beneath.

PASCAL

Pascal, a good idea, but marred in implementation. As Pascal was designed for the university set, it is not surprising that in its pure form it is of little use in real applications. To make the language more useful, various extensions have been added,



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

but these nice extensions have made Pascal nonportable. For a good laugh, compare Pascal/VS to PC/Pascal. For example, in PC/Pascal, metalanguage commands are placed inside Pascal comments, insuring a measure of portability. In Pascal/VS, compiler directives are not allowed in comments. There are other differences, too.

Pascal, like C, is a treasure chest for those who enjoy unbalanced structures. Who needs balanced structures with BEGIN and END statements?, you cry. Try nested loops for one.

Pascal is so prim and proper that, without extra effort, the "main" routine must appear as the last routine in a file. This insures that all procedures, functions, etc., have been defined. This is a great idea, but it conflicts with my personal taste. I have always liked to provide the "main" routine first and have subroutines appear afterward. What happened to personal preference? That's the Ctrl-breaks.

Pascal does have a crispness that C totally lacks. And the idea of having the compiler check the variable types for proper usage is extremely powerful. It saves programs (and programmers) from many obscure bugs. (Please, don't talk to me about LINT — that's the stuff you find in belly buttons.)

ADA

Recall the polka: "I don't want her, you can have her, she's too fat for me!" That seems to summarize the feeling of many programmers toward Ada. She is overweight, but has other redeeming charms. Ada has addressed many of the problems found in Pascal, maintained strong type checking, and Ada's "string-slicing" capabilities rival the power of BASIC string-handling.

Ada is not yet available for the PC, but will be someday. Why do I like Ada? Because legally no compiler can be called an Ada compiler unless it passes over 2000 source code tests. (So I have read.) And the legal rules are enforced by no less than the Department of Defense. Now that's backing from the top. This stranglehold will have a profound impact on the portability of Ada programs, making them highly standardized. Ada may be the last language that I will have to learn.

MY DEFENSE (INSANITY?)

Time is growing short and I had better wrap up my tirade.

Languages like FORTRAN, BASIC, and COBOL are kept alive because a great deal of code has been written in them and they are relatively easy to use. Simple programs can be written in those language without a great deal of forethought. One may even say that these languages are convenient to use. Maybe, just to make it a nicer world, we ought to have languages that are convenient to use and languages that are designed for programmers! Anyway, if you have a great desire to learn FORTRAN, BASIC, or COBOL I will gladly give you my old textbooks.

I like assembly language because it produces tiny, fast modules, which are a key ingredient in any user- or programmer-friendly product. (While the term "user-friendly" has been overused, the term "programmer-friendly" is hardly recognized or known.) I start to sweat when handed modules of 100K or larger in size. Why, you ask? Certainly virtual storage and faster RAM chips will solve all problems. No. Only the expenditure of further funds will make fat, oversize, slow modules run in an acceptable manner. Enter the PC—AT. Large programs are not user-friendly or programmer-friendly. Programming in assembly language is a painful experience, but the end result can be a slim, trim, powerful product that you would be proud to have created.

Furthermore, I find that to really get any useful code from a compiler, I constantly need to use assembly language. Consider interfacing to the ROM-BIOS or using the DOS EXEC call from a high-level language. It's "crying time down south" unless you know some assembly language, as high-level lan-

fileMASTER
THE DISK UTILITY
FINALLY!!
A Disk Utility for the IBM-PC that's fast and easy to use.
• FILE MODE
• DISPLAY
• PRINT
• MODIFY
• SECTOR MODE
• BROWSE
• SEARCH
• DOS 1.1 to 3.0
ONLY $39.95 + $2.00 SHIPPING, CA + 6.5% TAX
J. L. SCHULLER & ASSOCIATES (818) 366-6934
14800 Rinaldi St., Suite 6, Mission Hills, CA 91345

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

guages don't even know what a cursor is, let alone how to move one.

My C compiler includes some wonderful functions such as "showSP" or "showDS". These functions return the current value of the stack pointer and the data segment. Stack pointer, data segment? In a high-level language? A high-level language is designed to hide these things from the programmer, remember? I fear that high-level languages are not a great deal more portable than assembly language. You want further evidence? Consider, for example, the EXEC function call. As EXEC is an operating-system-dependent function, you compromise code portability any time you use this call, independent of the source code language used to write the module. As I said, in actual use, high-level languages are not really much more portable than assembly language, so you might as well go for the gusto. High-level languages are easier to use, and their source code is more manageable than assembly's. But speak not of portability.

Despite my criticisms of other languages, much of which was intended to be more humorous than serious, the bottom line is, How much time can you spend learning new languages? When I first entered the kingdom of computer languages, I was dazzled. The names of the languages dazzled me, the spectacular claims of their supporters dazzled me, the history of each language dazzled me. But when I cleared my eyes, I realized that I did not have enough time to become proficient in each of the many languages. Also, why learn Pascal when, in all likelihood, I will just scrap my source code in a short time? For surely a "better" language will be developed. Languages are in a state of transition and programmers are the pioneers. And I'm tired of being gunned down. One might as well use assembly language to take full advantage of the PC while the other languages vie for supremacy.

Even worse is that language X has children Y,Z,P,Q. It never seems to end, and it's time to practice birth control in the area of programming languages, as each language has its own curious syntax and rules. For this reason, I crave long-term standardization, and Ada is the first language that stands a ghost of a chance of remaining standard and faithful to its original charter. Until Ada, I want power and will use only assembly language.

If you think that this article is crazy, you ought to talk to me sometime! □

Now With Flipscreen

SYMD
SYMBOLIC•DEBUGGER

Faster, easier more productive

SYMD is a unique new programming aid that dramatically reduces the time and effort needed to identify and correct programming errors. SYMD simplifies debugging by utilizing source code symbols and line numbers wherever an address or offset is required. Flipscreen feature allows separate display, including graphics, for SYMD and program under test. SYMD also lets you: Examine and/or change real number data; assign permanent or temporary breakpoints, including pass counts to control looping; check instruction execution sequences with backtrace commands; use profiling commands to identify the most active parts of a program. And much more. SYMD can be used with compiler or assembly language programs. System requirements: PC-DOS™ or MS-DOS™ 1.1 or 2.0; 192K recommended; 80-column display.

SYMD is priced at only $125. Free literature on request. VISA or MasterCard accepted.

™MS-DOS is a trademark of Microsoft Corporation. ™PC-DOS is a trademark of the IBM Corporation.

D+V Systems
400 Amherst Street Nashua, NH 03063 (603) 881-7140

Famous Computer Personalities #25
G. Feldman Brechner, inventor of the first hard disk, was 52 years ahead of his time. The hard disk was a wonderful invention, but computers were yet to be invented.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BLAISE'S CTOOLS

by Rex Jaeschke

Price: CTOOLS $125
Blaise Computing Inc.
2034 Blake Street
Berkeley, CA 94704
(415) 540-5441

CTOOLS

C TOOLS is a library of functions callable from C programs written using the Microsoft/Lattice compiler (V1.4 and V2.0) and Computer Innovations C86 compiler (V1.33D and Optimizing Compiler V2.01A).

The installation kit includes 3 SS/DD diskettes (one each for source, object and libraries), a set of documentation in an 8 1/2' x 11' three-ring binder and a registration form. Registration entitles the user to receive technical support, a copy of the quarterly newsletter, BLAISE NEWS, and information about product updates.

To install CTOOLS you will need MS/PC-DOS V1.x (or later) and one of the compilers mentioned above. As all functions are supplied with source code (mostly in C, the rest in IBM/Microsoft assembler) it should not take much effort to use these tools with some other C compilers. (All C functions have been compiled with the DeSmet V2.3 development system producing only one or two minorproblems.) If your compiler does not generate IBM/Microsoft compatible assembly code, you will need to rewrite the BIOS interface. The hardware environment required by CTOOLS is that needed by your particular compiler. Certain hardware specific assembler routines require IBM-PC BIOS compatibility such as that found in systems from Compaq, Columbia and Corona.

If you have survived the installation and setup of your compiler, installing CTOOLS should be relatively straightforward. The installation batch procedures will only run under DOS V2.x. Due to their complexity and the fact that most if not all serious software developers have upgraded to this level, this is not an unreasonable approach. If you must install CTOOLS under DOS V1.1, read the instructions several times, review the V2.x batch procedures and draw up a plan of attack before you begin the installation proper. The user is assumed to be familiar with the C language, the PC/MS-DOS operating system and their particular compiler. Microsoft compiler users would find a knowledge of MS-LIB useful, particularly if they wish to replace library modules.

Some of the compilers supported allow various size program models which in turn require a different set of assembler macros and header files to be used. The copying and renaming of these model-dependent files is automatically handled by the installation batch procedures. Select the model you require with care, as once you have installed CTOOLS, you can only write programs of that model size. (Of course you could reinstall CTOOLS using a different model on another diskette or directory.)

Appendices C and D of the documentation describe in detail the use of the installation batch procedures and the method of compiling and linking user programs with both the CTOOLS and compiler libraries. A complete object library for each program model supported is provided in DOS LINK library format. Should you wish to modify a CTOOLS function, or write your own version of one, the necessary steps to do so are also provided. A useful C source code formatting program (CONCAT) is provided to allow you to get a printed listing of the library rouines.

The documentation set is very thorough and well laid out. It contains a chapter on each of the five groups of library functions, a quick reference guide, and a section that describes the four demonstration programs. The quick reference guide is part of the manual (not a separate card) and contains a brief yet adequate summary of each function in alphabetical order within a group. Each function summary contains the page number of the full description for that function. The absence of a cross-reference index is significantly offset by the table of contents, the general layout and the inclusion of the reference guide.

The chapter covering the demonstration programs contains nothing more than a listing of their source code. These programs are fairly straightforward, and you should compile and build them as an exercise to test if CTOOLS (and its use with your compiler) has been implemented properly. A batch procedure to do this would be a nice addition to the CTOOLS installation process.

The five function groups are 1) String, 2) Screen handling, 3) Utility, 4) Graphics interface and, 5) Applications. Each group has a two-character abbreviation which is used as a prefix in each function's name. For example, the string handling function that extracts the left n characters from a string is called STCLEFT. All functions have names of eight or less alphabetic

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

characters which allows their source code filenames to exactly match the function names.

The string functions include routines to extract various substrings, to replace characters or strings within other strings, to locate substrings and characters and to translate a string according to an edit mask. A string conversion routine STPCVT can be used to delete leading, trailing, or all whitespace; to convert to/from upper/lower case, and to discard all control characters. (This particular function is highlighted in one of the demonstration programs.) Other routines return the system date and time.

A useful discussion of the monochrome and color/graphics controllers introduces the chapter on screen handling. Functions are provided to set and get the video attributes, to switch between display pages, to scroll, to position the cursor, and to define the cursor size. Various utility functions provide random number generation and access to keyboard scan codes.

Several graphics primitives are provided to select palettes and colors, to put and get a point and to draw relative and absolute lines. No circle, box, painting, window, view or draw capabilities (such as in BASIC V1 and V2) are provided. Likewise, sound and play functions have been omitted.

The application functions provide some elementary data entry capabilities, can display an optional prompt and accept a single field. These routines are quite basic and anyone seriously interested in writing programs for data entry would do well to look at C VIEW MANAGER (also a Blaise Computing product), which was especially designed for that purpose.

One useful routine gives access to the command line used to invoke the program, exactly as it exists in the DOS Program Segment Prefix. This may be different from that obtained via the arguments argv and argc as their generation involves the removal of all white space.

While access to source code can be nice, this is highly dependent on its quality and the amount and level of comments included. CTOOLS function sources are well laid out and each has a lengthy introduction describing the purpose, philosophy and method of proper use. In some functions, this documentation is larger than the program itself. Liberal use is made of tabs, blanklines and blanks to make the format pleasing and easy to follow.

All pointer variables begin with the letter "p" and all function types are declared even if they are integer. One thing to watch is that some variable names (and compile-time constants) are longer than eight characters although there does not seem to be any duplication within those first eight between different variables.

One possible problem is that the null statement (particularly in FOR loops) is not put on a line by itself and is therefore harder to see. This may prove to be a source of error for those of you who modify any of the routines provided.

Individual source statements are well documented and comments seem to begin in a particular column which makes them line up. Annotating C statements can be a real problem, particularly if you stagger the start position of comments,but Blaise seems to have found a workable solution.

The assembler routines are of the same high standard as the C functions. However, the documentation concerning the batch files and procedures used to implement them could do with a little more explanation.

Without a doubt, CTOOLS includes some of the best written C and assembler source code you could hope to see. It appears that the designers and programmers have done their homework well and it is refreshing to see such a professional effort in a marketplace that all too often is burdened with less than mediocre products. Even without the source code, CTOOLS would be a bargain. □

(For programmers only...)

Stan's Editor

A full-screen editor for the PC.

Stan's Editor is small, fast, and powerful. It uses all available memory on the PC. (With a 512K PC, you can edit over 430K in memory!) The editor has a split-screen, so you can edit two files, each in a 12-line window, or edit a single file on the full screen. The editor allows very long (up to 65,000 bytes!) lines, with horizontal scrolling.

Stan's Editor has auto-indent (great for C and Pascal), and has a "match this brace (or parenthesis)" command. For assembler, the editor's condensed display shows only labels starting in column one, which provides a summary view of your code. DOS directory pathnames are supported and you can exit to DOS temporarily.

Stan's Editor is written in assembler. It's small (26K .COM file) and *very* fast! Three Help screens are built-in, giving you an online reference card. The editor also has optional word wrap. It is very easy to configure (you can even select your cursor).

Stan's Editor is only $65. Or for just $5.00 (applies toward purchase) our demo disk does everything except save a file.

Demo disk only $5

THE CODE WORKS
5266 Hollister, Suite 224
Santa Barbara, CA 93111
(805) 683-1585

Attention OEMs: we are dealing! Give us a call.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MISC.

SHRINK-WRAP YOURSELF

If you have a one-man software house and want to look like the big boys, get hold of one of these plastic shrink- wrappers and seal your software products in style. AJM sells a small table-top plastic sealer and industrial blow dryer that solves the problem. Under $300 for the whole shootin' match.

Shrink Wrap Kit
AJM Associates
5180 Meridian Ave.
San Jose, CA 95118
(408) 723-3701

GRAYSCALER FOR YOUR GREEN SCREENS

Connect your color graphics board to an inexpensive green or single-colored monitor and get the full gray spectrum for added crispness and legibility. If you are tired of playing games you just can't win or watching colored text you just can't read, try one of these.

GRAYSCALER ($59.95)
AOCADO Computer
17352 Yorkshire Ave.
Yorba Linda, CA 92686
(714) 528-1025

SOURCE CODE DEBUGGER FOR LATTICE

MSD has released a long-needed source code debugger for C programmers using Lattice. It is reasonably priced and offers a lot: breakpoints, single-stepping, online help screens, variable and register displays.

C Source Code Debugger ($165)
Micro-Software Developers, Inc.
214 1/2 W. Main St.
St. Charles, IL 60174
(312) 377-5151

SELF-HYPNOSIS WHILE YOU ARE PROGRAMMING

Subliminal suggestions can now be a daily part of your life. Pick a message like "I am a dynamite programmer" and have it permanently implanted in your brain by repeated subliminal blinking on the screen. The New Life Institute says this type of therapy is good for self-hypnosis and deep relaxation. (Ed: They need to sell some kids' games with this feature in them. We could clean up half of America this way.) For more info:

The Subliminal Hypnosis Package ($75)
New Life Institute / Software Division
P.O. Box 2390
Santa Clara, CA 95063
(408) 429-1122

INSTANT PHOTO-QUALITY HARD COPY FROM YOUR SCREEN

Mitsubishi is selling a hard-copy machine (P50U) that looks a little like a VCR box. You plug the box into your computer (or TV) and whenever you want a B&W print you punch the button. You can choose positive or negative images, and it only takes 20 seconds. The thermal prints are 4 x 3 1/2 inches. Ideal for enhancing your presentations and documentation or capturing your favorite football play from the screen. Cheap at $390.

P50U Video Printer
Mitsubishi
Contact: Phil Callahan
1-800-421-1132

DO-IT-YOURSELF BAR CODE PROGRAM

Percon, a quality bar code manufacturer, has introduced the ultimate bar code printing program. It prints the major codes that businesses are currently using: Code 39, Codabar, and I 2 of 5. Print the codes from a file or from keyboard input. Descriptors or additional text may be printed above or below the bar code. The package runs on the IBM PC and almost any dot-matrix printer.

WRITEBAR ($99)
Percon
2190 W. 11th St.
Eugene, OR 97402
(503) 344-1189

ADVERTISERS INDEX

- Blaise Computing 19
- Borland International . . . . . Cover/Back
- Creative Solutions 18
- Cybernetic Micro Systems 14
- D&V Systems 44
- Data Base Decisions 29
- David Smith Software 13
- DWB Associates 33
- Ecosoft, Inc. 39
- Essential Software 23
- Greenleaf Software 4
- J.L. Schuller & Associates 43
- Micro-Motion 28
- Morgan Computing 15
- O.E.S.-Systems 35
- Opt-Tech . . . . . 8 & 34
- Oregon Progressive Software 36
- Phoenix Computer Products . . . . . Cover/Inside Front
- Programmers Shop 7
- Qualitas 31
- Rational Systems, Inc. 10
- The Code Works 46

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ARE YOU AN ISSUE SHY OF A FULL SET?

Complete your library! Order any back issues you may have missed — or become a new subscriber. You'll be glad you did.

- Vol. 1, No. 1: Norton Pascal I/O, Program Style Manual, COBOL, System Analysis, BASIC modules, Rounding error.
- Vol. 1, No. 2: Defining character sets, C86 BIOS calls, Copy protection, Pascal code, Input routines, Writing an Assembler.
- Vol. 1, No. 3: ASM dumb terminal, C source code, ASM graphics line drawing, Text editors reviewed, Utilities.
- Vol. 1, No. 4: Diskette directory via Pascal, MASM WHILE-WEND, C/ASM interface, ASM stay resident routine, Structured programming.
- Vol. 1, No. 5: Multi-tasking software review, Transporting C Programs, Programmers calculator, dBase bargraphs, Utilities.
- Vol. 1, No. 6: ASM printer configuration, DOS patch utility, Hi-speed graphics, Software ownership, dWINDOW review.
- Vol. 2, No. 1: Freelance consulting, Sorts in C, Pascal text editor source code, Designing national database, Ultra utilities.
- Vol. 2, No. 2: Understanding C through Assembly language, Utility routines in C and Assembly, Music in C (using a C music program), Cornucopia of software development tools.
- Vol. 2, No. 3: Software development environments, Creating a productive screen handling tool, Generating graphics within windows, Create, edit and play your musical ideas, Linking Assembler to Pascal and Basic.
- Vol. 2, No. 4: Writing a custom ROM BIOS for PC-DOS, The future of disk protection, Time-saving MASM Macros, C routines you can purchase.
- Vol. 2, No. 5: Scroll control using BASIC and Assembler, RTool, a PC-resident programmer's toolkit, Calling Assembly routines from dBase II, Pascal's pointer power.
- Vol. 2, No. 6: Periscope-Building a hardware board for the IBM PC, AT&T surprises all, dBase II to dBase III?, Intel 80286 bugs by FNE, calendar functions for BASIC.
- Vol. 3, No. 1: November's Comdex, C - its uses and abuses, Using the C preprocessor, Memory segmentation examples, Index files (part 1), Stuctured programming language, Periscope, a symbolic debugger.

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Telephone _____

Send me the following back issues: (Circle the issue(s) desired)
$5.50 each U.S., ($6.50 Canada, $8.50 Foreign, U.S. funds) pre-paid only

Vol. 1, No. 1	Vol. 1, No. 5	Vol. 2. No. 3	Vol. 3, No. 1
Vol. 1, No. 2	Vol. 1, No. 6	Vol. 2. No. 4	
Vol. 1, No. 3	Vol. 2, No. 1	Vol. 2. No. 5	
Vol. 1, No. 4	Vol. 2, No. 2	Vol. 2. No. 6	

☐ 1 year bi-monthly subscription:
$24.00 ($30 Canada, $40 Foreign, U.S. funds only)

☐ Please renew my subscription

☐ Check enclosed ☐ Bill me (subscriptions only)

☐ Charge my Bankcard: ☐ VISA ☐ MC ☐ Am Ex

_____ Expir. Date _____

Signature _____

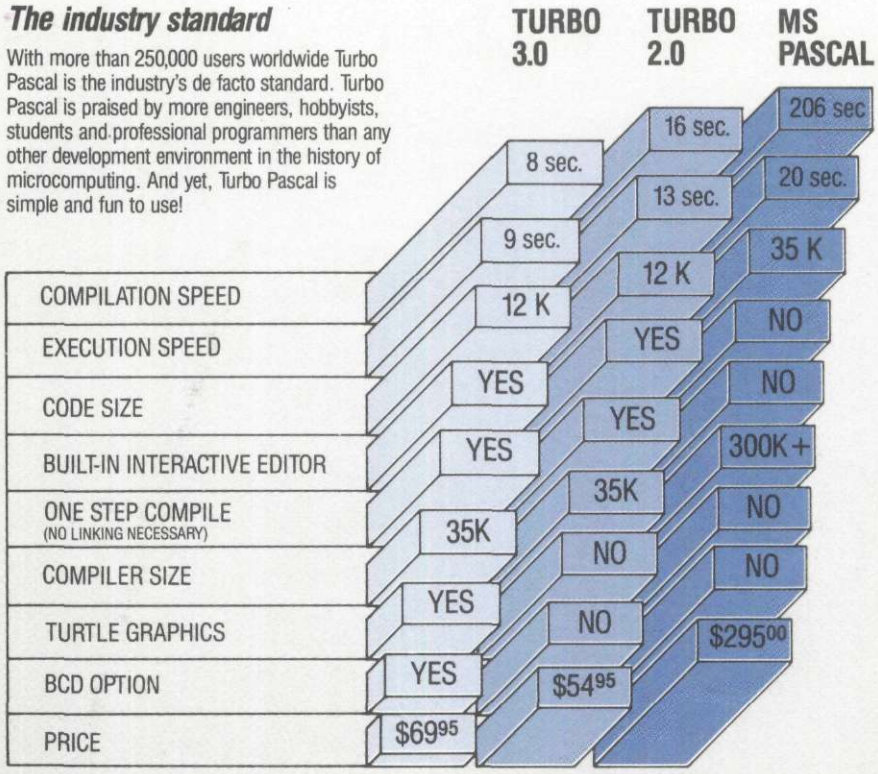
Send to:
PROGRAMMER'S JOURNAL
P.O. Box 3000, Dept. EE
Denville, NJ 07834
(503) 345-3043

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

They said it couldn't be done. Borland Did It. Turbo Pascal 3.0

The industry standard

With more than 250,000 users worldwide Turbo Pascal is the industry's de facto standard. Turbo Pascal is praised by more engineers, hobbyists, students and professional programmers than any other development environment in the history of microcomputing. And yet, Turbo Pascal is simple and fun to use!



The best just got better: Introducing Turbo Pascal 3.0

We just added a whole range of exciting new features to Turbo Pascal:

- First, the world's fastest Pascal compiler just got faster. Turbo Pascal 3.0 compiles twice as fast as Turbo Pascal 2.0! No kidding.
- Then, we totally rewrote the file I/O system, and we also now support I/O redirection.
- For the IBM PC versions, we've even added "turtle graphics" and full tree directory support.
- For all 16 Bit versions, we now offer two additional options: 8087 math coprocessor support for intensive calculations and Binary Coded Decimals (BCD) for business applications.
- And much much more.

The Critics' Choice.

Jeff Duntemann, PC Magazine: *"Language deal of the century . . . Turbo Pascal: It introduces a new programming environment and runs like magic."*

Dave Garland, Popular Computing: *"Most Pascal compilers barely fit on a disk, but Turbo Pascal packs an editor, compiler, linker, and run-time library into just 29K bytes of random-access memory."*

Jerry Pournelle, BYTE: *"What I think the computer industry is headed for: well documented, standard, plenty of good features, and a reasonable price."*

Portability

Turbo Pascal is available today for most computers running PC DOS, MS DOS, CP/M 80 or CP/M 86. A XENIX version of Turbo Pascal will soon be announced, and before the end of the year, Turbo Pascal will be running on most 68000 based microcomputers.

An Offer You Can't Refuse

Until June 1st, 1985, you can get Turbo Pascal 3.0 for only $69.95. Turbo Pascal 3.0, equipped with either the BCD or 8087 options, is available for an additional $39.95 or Turbo Pascal 3.0 with both options for only $124.95. As a matter of fact, if you own a 16 Bit computer and are serious about programming, you might as well get both options right away and save almost $25.

Update policy

As always, our first commitment is to our customers. You built Borland and we will always honor your support.

So, to make your upgrade to the exciting new version of Turbo Pascal 3.0 easy, we will accept your original Turbo Pascal *disk* (in a bend-proof container) for a trade-in credit of $39.95 and your Turbo87 original *disk* for $59.95. This trade-in credit may only be applied toward the purchase of Turbo Pascal 3.0 and its additional BCD and 8087 options (trade-in offer is only valid directly through Borland and until June 1st, 1985).

(*) Benchmark run on an IBM PC using MS Pascal version 3.2 and the DOS linker version 2.6. The 179 line program used is the "Gauss-Seidel" program out of Alan R. Miller's book: *Pascal programs for scientists and engineers* (Sybex, page 128) with a 3 dimensional non-singular matrix and a relaxation coefficient of 1.0.

NOT COPY-PROTECTED

TURBO PASCAL

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133

YES! I want the Best!

Please send: Quantity

Pascal 3.0 $ 69.95 ____

Pascal w/8087 $109.90 ____

Pascal w/BCD $109.90 ____

Pascal w/8087 & BCD $124.95 (SAVE $24.90) ____

*These prices include shipping to all U.S. cities. All foreign orders add $10 per product ordered.

Trade-in Credit Claimed: ____

Amount: (CA 6% tax) ____

Payment: VISA MC BankDraft Check

Credit Card Expir. Date: ____/____

Name on Card: ____

Card #: ____

Carefully Describe your Computer System!

Mine is: ___ 8 bit ___ 16 bit
I Use: ___ PC-DOS ___ MS-DOS
___ CP/M 80 ___ CP/M 86

My computers' name/model is: ____

The disk size I use is:

☐ 5¼" ☐ 8" ☐ 3½"

For update: original Turbo disk must accompany order

Name: ____

Shipping Address: ____

City: ____ Zip: ____

State: ____

Telephone: ____

COD's and Purchase Orders WILL NOT be accepted by Borland. California residents: add 6% sales tax. Outside USA: add $10 and make payment by bank draft, payable in US dollars drawn on a US bank.

3

BORLAND INTERNATIONAL

Software's Newest Direction
4113 Scotts Valley Drive
Scotts Valley, California 95066
TELEX: 172373

Turbo Pascal is a registered trademark of Borland International, Inc.

PRODUCED 2004 BY UMZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED